

BÀI TẬP PYTHON

(Phần cơ bản)



LÊ VĂN HẠNH
levanhanhvn@gmail.com
THÁNG 9-2019

MỤC LỤC

1	PHẦN CƠ BẢN	2
1.1.	Các lệnh điều khiển cơ bản	2
1.2.	Strings.....	13
1.3.	Bitwise Operator.....	19
2	USER DEFINE FUNCTION - MODULE - PACKAGE	20
2.1.	Xây dựng hàm	20
2.2.	Anonymous function (hàm ẩn danh)	21
2.3.	Xây dựng hàm đệ quy (Recursion).....	23
2.4.	Datetime module	25
3	CÁC ĐỐI TƯỢNG DẠNG DANH SÁCH TRONG PYTHON (<i>Iterator object or Sequences types</i>)	26
3.1.	Lists	26
3.2.	Tuple.....	35
3.3.	Dictionary.....	36
3.4.	Set.....	42
3.5.	Phối hợp các đối tượng dạng sequence type	43
3.6.	Sử dụng hàm map để chuyển đổi giữa các đối tượng dạng danh sách.....	45
3.7.	Xây dựng hàm ẩn danh (<i>Anonymous Function</i>) cho Iterator object.....	46
3.8.	Sử dụng kỹ thuật Comprehension cho cho Iterator object	48
3.9.	Giải quyết 1 yêu cầu bằng cả hai kỹ thuật Comprehension & Anonymous Function	52
4	XỬ LÝ NGOẠI LỆ (Exception Handling)	54
5	FILE - DIRECTORY - OPERATING SYSTEM	55
5.1.	Text file	55
5.2.	CSV file.....	57
5.3.	Directory (Thư mục)	60
5.4.	Khác.....	61
6	CLASS.....	63
6.1.	Class & Object.....	63
6.2.	Kế thừa (Inheritance).....	65
6.3.	Abstract	66
7	DATA STRUCTER.....	67
7.1.	Enum	67
7.2.	Array.....	67
7.3.	Search & sort in list	67
7.4.	Queue & Stack.....	68
7.5.	Linked list.....	69
7.6.	Binary search tree	70

PHẦN CƠ BẢN

1.1. Các lệnh điều khiển cơ bản

1.1.1. Các lệnh xuất nhập - Biến – Toán tử

1.1.1.1. Sử dụng hàm print

1/. Sử dụng lệnh print để lần lượt in ra các hình sau:

a.-	b.-	c.-	* * *	d.-	* * * * *
* * * * *	* * * * *	* * *		* * *	
* * * * *	* * * * *	* * *		* * *	
* * * * *	* * * * *	* * *		* * *	

2/. Sử dụng lệnh print để in ra màn hình chữ PYTHON như hình minh họa sau:

```

XXXXX XX XX XXXXXX XX XX XXXX XX XX
XX XX X X XX XX XX XX XX XXX XX
XXXXX X X XX XXXXXX XX XX XXXXXX
XX XX XX XX XX XX XX XXX XX
XX XX XX XX XX XXXX XX XX

```

3/. Chỉ sử dụng 1 lệnh print, yêu cầu in ra màn hình đoạn thơ Quê hương với định dạng như hình minh họa.

☞ Gợi ý: sử dụng ký tự tab ('\t') để lùi các đầu dòng sang phải.

QUÊ HƯƠNG

. . .

Quê hương là chùm khế ngọt

Cho con trèo hái mỗi ngày

Quê hương là đường đi học

Con về rợp bướm vàng bay

. . .

1.1.1.2. Sử dụng hàm input

4/. Cho nhập 2 số nguyên a, b. In ra tổng của 2 số a và b như minh họa sau:

Ví dụ: a=3, b=5 \Rightarrow in ra 3 + 5 = 8

Giáp Văn Thạch

a. Cho nhập 2 số bằng lệnh riêng biệt

b. Cho nhập 2 số cách nhau bởi dấu phẩy (,) trên cùng một dòng lệnh bằng cách sử dụng phương thức split của kiểu dữ liệu chuỗi

```
a, b = input("...").split()
```

c. Sử dụng hàm map(), phương thức split của dữ liệu kiểu chuỗi để cho nhập 1 lần nhiều giá trị, các giá trị này được ngăn cách theo chỉ định trước của chương trình (khoảng trắng, dấu phẩy hoặc bất kỳ ký tự nào)

Ví dụ: sử dụng phối hợp 2 hàm map và int với hàm input để cho nhập 2 số cách nhau bởi dấu chấm phẩy:

```
a, b = map(int, input("...").split(';'))
```

5/. Cho nhập 2 số nguyên a, b trên cùng 1 dòng (cách nhau bởi khoảng trắng). In ra kết quả của a lũy thừa b (a^b) ra màn hình dưới dạng "%d^%d=%d".

6/. Cho nhập số nguyên dương a có giá trị từ 1 đến 9. In ra tổng của a+aa+aaa.

Ví dụ: a=5 \Rightarrow in ra 5 + 55 + 555 = 615

📖 Gợi ý: sử dụng phép nối chuỗi các số nguyên (vd: với a=5: "%s%s" % (a,a) => 55) và hàm int (đổi kiểu dữ liệu từ chuỗi sang số nguyên).

- 7/. Viết chương trình Python để chứng minh rằng hai biến chuỗi có cùng giá trị trở đến cùng một vị trí bộ nhớ bằng cách khai báo biến `x = "Python"`, `y = "Python"`, `z = "python"`. In ra địa chỉ bộ nhớ của các biến

📖 Gợi ý: □ Sử dụng hàm `id (tên_biến)` để lấy địa chỉ của biến được lưu trong bộ nhớ.
 □ Sử dụng hàm `hex` để chuyển giá trị số đang có sang số hệ thập lục.

1.1.1.3. Sử dụng phương thức format của chuỗi có trong print

- 8/. Cho nhập 3 số nguyên a, b, c. In ra 3 số trên theo thứ tự tăng dần. Yêu cầu sử dụng phương thức format trong việc xuất kết quả, ví dụ:

```
print('{} {} {}'.format(biến1, biến2, biến3))
hoặc print('{0} {1} {2}'.format(biến1, biến2, biến3))
```

📖 Gợi ý: dùng hàm `min`, `max` để tìm số nhỏ, lớn nhất, từ đó suy ra số lớn nhì. Sau đó dùng phương thức format của dữ liệu kiểu chuỗi để xuất kết quả.

- 9/. Cho nhập chiều dài, chiều rộng và chiều cao của hình khối chữ nhật (theo cm). Tính và xuất kết quả theo ví dụ sau:

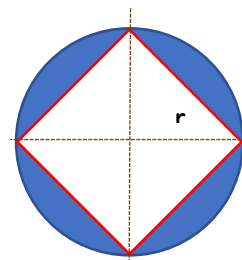
```
Nhập chiều dài đáy hình khối chữ nhật (cm):>? 2.134
Nhập chiều rộng đáy hình khối chữ nhật (cm):>? 3.4567
Nhập chiều cao hình khối chữ nhật (cm):>? 4.1
Số lượng số lẻ cần hiển thị:>? 2
Diện tích đáy hình chữ nhật = 7.38cm2
Thể tích hình khối= 30.24cm3
```

📖 Gợi ý: để in các ký tự số mũ, sử dụng mã UNICODE lần lượt là `\u00b2` và `\u00b3` trong lệnh `print`.

1.1.1.4. Tính toán đơn giản

- 10/. Ban đầu trong rừng có 1 cặp thỏ, biết rằng cứ sau 1 tháng thì số lượng thỏ trong rừng tăng lên gấp đôi. Hỏi sau m tháng thì có bao nhiêu con thỏ (giả sử các con thỏ không chết).

- 11/. Cho hình vuông nội tiếp hình tròn như hình minh họa bên cạnh. Viết chương trình cho nhập bán kính hình tròn (r) là 1 số thực có giá trị lớn hơn 0, tính phần diện tích có màu đậm trong hình.



- 12/. Tạo chương trình yêu cầu người dùng nhập tên và tuổi của họ. Giả sử năm hiện tại là 2020, chương trình in ra màn hình chuỗi (s) thông báo năm mà người đó tròn 100 tuổi.

Ví dụ: Nhập tên: Tý
 Nhập tuổi: 21
 Đến năm 2099, bạn Tý sẽ tròn 100 tuổi

- 13/. Có 9 loại tiền 1, 2, 5, 10, 20, 50, 100, 200, 500. Cho nhập số tiền X. Chuyển số tiền X ra các loại tiền sao cho số lượng là ít nhất. In ra số tờ tiền của mỗi loại, tổng số tờ tiền của tất cả các loại.

Ví dụ với X=1234, sẽ in ra:

```
So tien 1234 duoc doi thanh:
Loai 500 gom 2 to
Loai 200 gom 1 to
Loai 100 gom 0 to
Loai 50 gom 0 to
Loai 20 gom 1 to
Loai 10 gom 1 to
Loai 5 gom 0 to
```

Loại 2 gồm 2 to
 Loại 1 gồm 0 to
 TỔNG CỘNG CÓ 7 TỜ

1.1.1.5. Đọc chương trình xác định kết quả

14/. Đọc chương trình xác định kết quả xuất ra của đoạn chương trình dưới đây. Sau khi ghi kết quả vào các ô tương ứng, sinh viên hãy viết và chạy đoạn chương trình này để kiểm tra lại:

a.		b.	
Mã lệnh	Kết quả	Mã lệnh	Kết quả
X= 5 print('X=', X)	X=5	x = True y = False print('x and y :',x and y)	x and y : False
X-= 1 print('X=', X)	X=4	print('x or y :',x or y)	x or y : True
X+= 3 print('X=', X)	X=7	print('not x :',not x)	not x : False
X= - X print('X=', X)	X=-7	print('x is y :', x is y)	x is y : False
X= True print('not X=', not X)	not X=False	print('x is not y :', x is not y)	x is not y : True

c.		d.	
Mã lệnh	Kết quả	Mã lệnh	Kết quả
X = 1 + 2 print('X=', X)	X= 3	x = 10 y = 4 print('x = %d, y = %d'%(x,y))	x = 10, y = 4
Y = X X = X- 1 print('X=', X)	X= 2	z = x==y print('x==y is', z)	x==y is False
Y = X X = X* 2 Y = X print('X=', X)	X= 4	z = x!=y print('x!=y is', z)	x!=y is True
X = X** 3 Y = X print('X=', X)	X= 64	z = x>y print('x>y is', z)	x>y is True
X = X+ 8 Y = X print('X=', X)	X= 72	x = 8 y = 9 print('x = %d, y = %d'%(x,y))	x = 8, y = 9
X= X% 7 Y= X print('X=', X)	X= 2	z = x>=y print('x>=y is', z)	x>=y is False
X= X// 2 Y= X print('X=', X)	X= 1	z = x<y print('x<y is', z)	x<y is True
		z = x<=y print('x<=y is', z)	x<=y is True

e.

Mã lệnh	Kết quả
x = 15 y = 12 print('Binary of x =', bin(x))	Binary of x = 0b1111
print('Binary of y =', bin(y))	Binary of y = 0b1100
print('x & y =', bin(x & y))	x & y = 0b1100
print('x y =', bin(x y))	x y = 0b1111
print('x ^ y =', bin(x ^ y))	x ^ y = 0b11
print('~x =', bin(~x))	~x = -0b10000
print('x << 2 =', bin(x << 2))	x << 2 = 0b111100
print('y >> 2 =', bin(y >> 2))	y >> 2 = 0b11

1.1.2. Cấu trúc điều kiện

15/. Cho người dùng nhập 1 số nguyên (n). Thực hiện:

- Cho biết n là số chẵn hay số lẻ
- Cho biết n có chia chẵn cho 4 hay không? Nếu không chia chẵn cho 4 thì n có chia chẵn cho 2 hay không?
- Gọi số n đã nhập là số bị chia. Cho người dùng nhập thêm 1 số nguyên khác (m) đại diện cho số chia. Cho biết n có chia chẵn cho m hay không?

16/. Thực hiện lại bài tập **13** với yêu cầu chỉ in những loại tiền có số tờ lớn hơn 0. (Có 9 loại tiền 1, 2, 5, 10, 20, 50, 100, 200, 500. Cho nhập số tiền X. Chuyển số tiền X ra các loại tiền sao cho số lượng là ít nhất.)

Ví dụ với X=1234, sẽ in ra:

```
So tien 1234 duoc doi thanh:
Loai 500 gom 2 to
Loai 200 gom 1 to
Loai 100 gom 0 to  #không in dòng này vì số lượng =0
Loai 50 gom 0 to   #không in dòng này vì số lượng =0
Loai 20 gom 1 to
Loai 10 gom 1 to
Loai 5 gom 0 to    #không in dòng này vì số lượng =0
Loai 2 gom 2 to
Loai 1 gom 0 to    #không in dòng này vì số lượng =0
TỔNG CỘNG CÓ 7 TỜ
Tong so loai = 5    #in thêm dòng này trong kết quả
```

17/. Cho người dùng nhập điểm (từ 0-100). Nếu điểm nhập có giá trị <0 hoặc >100, chương trình in ra câu báo lỗi '**Chỉ nhận các giá trị từ 0 đến 100**' và kết thúc chương trình. Ngược lại, in ra xếp loại của điểm số đó. Quy định xếp loại được cho trong bảng bên:

Điểm số	Xếp loại
>=90	A
Từ 80 đến 89	B
Từ 70 đến 79	C
Từ 65 đến 69	D
Dưới 65	E

18/. Viết chương trình cho nhập số nguyên dương n (có giá trị trong khoảng từ 1 đến 10. In ra chữ số La mã tương ứng.

19/. Viết chương trình kiểm tra giúp người dùng tự kiểm tra sức khỏe dựa trên chỉ số BMI (Body Mass Index - chỉ số khối cơ thể) bằng cách cho người dùng nhập vào cân nặng (tính theo kg) và chiều cao (tính theo mét) của họ, chương trình xuất ra kết quả đánh giá. Biết rằng:

- Công thức tính BMI: **cân nặng (kg) / (chiều cao (m) * chiều cao (m))**
- Đánh giá BMI của người Việt Nam được cho trong bảng sau:

BMI	Đánh giá
<18.5	Thiếu cân (Underweight)
Từ 18.5 đến 22.99	Bình thường (Normal)
Từ 23 đến 24.99	Thừa cân (Overweight)
Từ 25 trở lên	Béo phì (Obese)

20/. Viết chương trình cho nhập số dương n . Chương trình thực hiện việc trừ n đi số x (với x là tổng các chữ số thành phần đang có của n). Chương trình thực hiện tìm n sao cho n nhỏ nhất và $n > 0$.

Ví dụ: với $n = 37$, sẽ in ra $n=9$ là kết quả cần tìm (vì $37-10=27$
 $27-9=18$
 $18-9=9$)

Tương tự với $n = 6$, sẽ in ra $n=6$ là kết quả cần tìm

21/. Cho nhập 2 số nguyên a, b trên cùng 1 dòng (cách nhau bởi khoảng trắng). Viết chương trình giải phương trình bậc 1: $ax + b = 0$.

22/. Cho nhập 3 số nguyên a, b, c trên cùng 1 dòng (cách nhau bởi khoảng trắng). Viết chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$.

23/. Một điểm trong mặt phẳng được biểu diễn bằng 2 tọa độ x và y . Cho nhập tọa độ 3 đỉnh A, B, C của 1 tam giác, cho nhập tọa độ điểm P . Xác định xem điểm P nằm trong hay nằm ngoài tam giác.

🔗 Gợi ý:

- Thực hiện tính 3 giá trị $z1, z2, z3$ theo công thức sau:

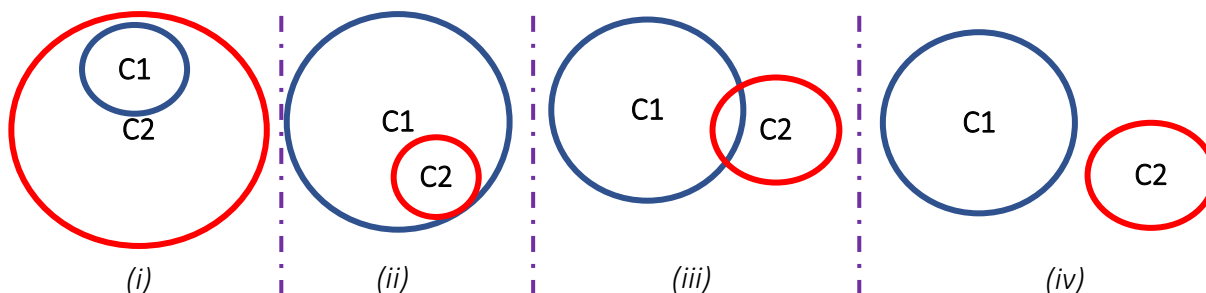
$$z1 = (xB - xA) * (yP - yA) - (yB - yA) * (xP - xA)$$

$$z2 = (xC - xB) * (yP - yB) - (yC - yB) * (xP - xB)$$

$$z3 = (xA - xC) * (yP - yC) - (yA - yC) * (xP - xC)$$

- Kiểm tra nếu cả 3 giá trị $z1, z2, z3$ cùng dương hoặc cùng âm thì điểm P nằm trong tam giác ABC, ngược lại là nằm ngoài.

24/. Mỗi vòng tròn trong mặt phẳng được xác định bởi các yếu tố tâm vòng tròn (x, y) và bán kính (R) . Viết chương trình cho nhập thông tin về 2 vòng tròn $C1$ và $C2$. Chương trình cho biết $C1$ và $C2$ thuộc trường hợp nào trong các trường hợp sau:



- (i)- $C1$ nằm trong $C2$
- (ii)- $C2$ nằm trong $C1$
- (iii)- $C1$ và $C2$ cắt nhau
- (iv)- $C1$ và $C2$ không cắt nhau (không có phần giao nhau)

25/. Nhập vào 6 số thực a, b, c, d, e, f cách nhau bởi dấu chấm phẩy (;). Giải hệ phương trình sau:

$$\begin{cases} ax+by=c \\ dx+ey=f \end{cases}$$

↪ Gợi ý: tính

$$D = \begin{vmatrix} a & b \\ d & e \end{vmatrix} = ae - bd ; \quad Dx = \begin{vmatrix} c & b \\ f & e \end{vmatrix} = ce - bf ; \quad Dy = \begin{vmatrix} a & c \\ d & f \end{vmatrix} = af - dc$$

Nếu $D \neq 0 \Rightarrow x = Dx/D; y = Dy/D$

Ngược lại nếu $Dx = 0$ hoặc $Dy = 0 \Rightarrow$ PT vô nghiệm

Ngược lại, \Rightarrow PT vô định

26/. Cho nhập 3 số nguyên a, b, c. Cho biết thứ tự của 3 số a, b, c có tạo thành 1 cấp số cộng hay 1 cấp số nhân hoặc không là cả 2 loại này. Nếu là cấp số cộng hay 1 cấp số nhân, hãy in ra số kế tiếp của dãy đó.

↪ Nhắc lại:

- Một cấp số cộng (*Arithmetic Progression* hoặc *Arithmetic Sequence*) là một dãy số thỏa mãn điều kiện: hai phần tử liên tiếp nhau sai khác nhau một hằng số gọi là công sai (*common difference*). Chẳng hạn, dãy số 3, 5, 7, 9, 11, ... là một cấp số cộng với công sai là 2.
- Một cấp số nhân (*Geometric Progression* hoặc *Geometric Sequence*) là một dãy số thỏa mãn điều kiện kể từ số hạng thứ hai, mỗi số hạng đều là tích của số hạng đứng ngay trước nó với một số không đổi. Hằng số này được gọi là công bội (*common ratio*) của cấp số nhân. Chẳng hạn, dãy số 5, 15, 45, 135, ... là một cấp số nhân với công bội là 3.

Ví dụ:

Nhập 1,2,3 in ra AP sequence. Next number of the sequence: 4

Nhập 2,6,18 in ra GP sequence. Next number of the sequence: 54.0

Nhập 1,3,2 in ra Not an Arithmetic Progression and Geometric Progression sequence

1.1.3. Cấu trúc lặp

27/. Cho nhập số nguyên n.

- a.- In ra các số từ 1 đến n.
- b.- In ra các số nhỏ hơn n và là bội số của 5.
- c.- In ra tổng các số từ 1 đến n.
- d.- In ra tổng các số chẵn nhỏ hơn n.
- e.- In ra các ước số chẵn của n và tổng các ước số chẵn đó.
- f.- In ra số chẵn nhỏ hơn và gần với n nhất.
- g.- Tìm ước số lẻ lớn nhất của n (nhỏ hơn n)

Ví dụ Nhập n=27, in ra Ước số lẻ lớn nhất của 27 là 9.

28/. Cho nhập 3 số nguyên n, số bắt đầu a và công sai d (*common difference*), trong đó n phải >0. In lên màn hình n số hạng đầu tiên của cấp số cộng (*Arithmetic Progression* hoặc *Arithmetic Sequence*).

29/. Cho nhập 3 số nguyên n, số bắt đầu a và công bội q (*common ratio*), trong đó n phải >0. In lên màn hình n số hạng đầu tiên của cấp số nhân (*Geometric Progression* hoặc *Geometric Sequence*).

30/. Cho nhập 2 số nguyên dương n và m ($n > m > 0$), liệt kê các ước số chẵn của n có giá trị nhỏ hơn m . Nếu không có ước số nào thỏa điều kiện nhỏ hơn m , chương trình cần in ra thông báo “không có ước số nào của n nhỏ hơn m ”.


Ví dụ Nhập $n=30, m=10$;

in ra các ước số của 30 nhỏ hơn 10 là 1, 2, 5, 6.


31/. Cho nhập n . In bảng cửu chương n theo hình minh họa bên (giả sử với $n=5$)

5	X	1	=	5
5	X	2	=	10
5	X	3	=	15
...
5	X	10	=	50

32/. Cho nhập 2 số nguyên a, b trên cùng 1 dòng (cách nhau bởi dấu phẩy – ‘,’). In ra các bảng cửu chương từ a đến b (khi $a < b$) hoặc từ b đến a (khi $b < a$).

 **Gợi ý:** hoán vị giá trị của a và b khi $b < a$.

33/. Cho nhập 2 số nguyên a, b ($a < b$). In ra các số nằm trong khoảng từ a đến b và bỏ qua các số chia hết cho 3 thuộc đoạn $[a, b]$.

 **Yêu cầu:** thực hiện bằng 2 cách:

- Cách 1: dùng lệnh (statement) *if*.
- Cách 2: dùng lệnh *continue*.

34/. Tính tổng và tích các chữ số của số nguyên dương n .

Ví dụ: $n=5084 \Rightarrow$ in ra tổng=17, tích=0

35/. Tìm chữ số lớn nhất có trong số nguyên dương n .

Ví dụ: $n=5084 \Rightarrow$ in ra số lớn nhất=8

36/. Cho nhập số nguyên dương n . Đếm số lượng chữ số chẵn, số chữ số lẻ có trong n .

Ví dụ: $n=5084 \Rightarrow$ in ra số lượng số lẻ=1, số lượng số chẵn=3

37/. Số may mắn: giả sử người ta cho rằng 1 số gọi là số may mắn nếu chỉ chứa toàn các số 6 hoặc số 8. Viết chương trình cho nhập số nguyên n , xét xem n có là số may mắn hay không?

Ví dụ: $n=686 \Rightarrow$ 686 là số may mắn.

$n=68626 \Rightarrow$ 68626 KHÔNG phải số may mắn.

38/. Cho nhập số nguyên dương n . Kiểm tra xem các chữ số của n có toàn lẻ (hay toàn chẵn) không?

39/. Cho nhập số nguyên n . In ra cách đọc của số n .

Ví dụ: Nhập $n = -105$. In ra màn hình: am mot khong nam.

40/. Cho nhập số nguyên dương n ($n \geq 2$). Hãy phân tích n thành tích các thừa số nguyên tố.


Ví dụ: nhập $n=1350$. In ra **1350 = 2 * 3 * 3 * 3 * 5 * 5**

41/. Cho nhập số nguyên dương n . In ra màn hình cách phân tích n thành thừa số nguyên tố nhưng dưới dạng số mũ.

Ví dụ: nhập $n=1350$. In ra **1350 = 2 * 3^3 * 5^2**


42/. Cho nhập số nguyên dương n ($11 \leq n \leq 100$), nếu nhập sai, yêu cầu nhập lại cho đến khi nhập đúng. Sau khi nhập đúng, chương trình in ra các số X nằm trong khoảng từ 11 đến n sao cho các ký số có trong X đều giống nhau.

Ví dụ: $n=50 \Rightarrow$ in ra 11, 22, 33, 44

 **Mở rộng:** cho trường hợp $n \leq 1.000.000$

43/. Cho nhập 1 số nguyên n ($n > 1$), nếu nhập sai, yêu cầu nhập lại cho đến khi nhập đúng. Sau khi nhập đúng, chương trình in ra chuỗi số *Fibonacci* từ 0 đến $\leq n$.

Ví dụ: với $n = 50$, sẽ in ra 0 1 1 2 3 5 8 13 21 34

 Nhắc lại: chuỗi số *Fibonacci* luôn bắt đầu từ 2 số 0 và 1. Mỗi số tiếp theo được xác định bằng cách cộng hai số liền trước nó.

44/. Viết chương trình trò chơi “Bao-Búa-Đinh” cho 2 người chơi. Mỗi người dùng được chọn 1 trong 3 món Bao, búa, đinh. Chương trình cho biết người thắng cuộc theo quy ước:

- Bao thắng Búa
- Búa thắng Đinh
- Đinh thắng Bao

a. Mở rộng 1: Sau mỗi lần thông báo kết quả, chương trình cho người dùng được chơi tiếp hay không? Nếu người dùng chọn chơi tiếp thì chương trình cho lặp lại việc chọn món có trong trò chơi.

b. Mở rộng 2: Khi kết thúc, chương trình cho biết số lần thắng của mỗi người chơi

45/. Cho nhập n . Tính tổng S (hoặc tích P) trong các trường hợp sau. Yêu cầu thực hiện:

- Tạo menu cho bài tập có dạng như hình sau. Tùy vào chức năng được chọn, chương trình sẽ gọi hàm thực hiện tính toán tương ứng.
- Sử dụng lệnh *for* và hàm *range()* để thực hiện tính toán.
- Mở rộng: sử dụng lệnh *while* để thực hiện tính toán thay cho lệnh *for*.

***** CHƯƠNG TRÌNH TÍNH TỔNG TÍCH *****

- 0.- Nhập n
 - 1.- Tính $S = 1 + 2 + 3 + \dots + n$
 - 2.- Tính $S = 1 + 3 + 5 + \dots + (2n+1)$
 - 3.- Tính $S = 1 + 1/2 + 1/3 + \dots + 1/n$
 - 4.- Tính $S = 1/2 + 1/3 + 1/4 + \dots + 1/(n+1)$
 - 5.- Tính $S = 1/2 + 1/4 + 1/6 + \dots + 1/(2n)$
 - 6.- Tính $S = (1) + (1+2) + (1+2+3) + \dots + (1+2+3+\dots + n)$
 - 7.- Tính $P = 1 * 2 * 3 * \dots * n$
 - 8.- Tính $P = (1) * (1+2) * (1+2+3) * \dots * (1+2+3+\dots + n)$
 - 9.- Kết thúc chương trình
- Chọn chức năng cần thực hiện (0-9): _

a. $S = 1 + 2 + 3 + \dots + n$

b. $S = 1 + 3 + 5 + \dots + (2n+1)$

c.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

d.

$$S = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n+1}$$

e.

$$S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \dots + \frac{1}{2n}$$

f. $S = (1) + (1+2) + (1+2+3) + (1+2+3+4) + \dots + (1+2+3+\dots + n)$

g. $P = 1 * 3 * 5 * \dots * (2*n-1)$

h. $P = (1) * (1*2) * (1*2*3) * (1*2*3*4) * \dots * (1*2*3*\dots * n)$

🔗 **Minh họa:** màn hình menu có dạng như sau:

```

          CHUONG TRINH IN TAM GIAC VUONG CAN
0.- Nhap canh cua tam giac
1.- Hinh 1.
2.- Hinh 2.
3.- Hinh 3.
4.- Hinh 4.
5.- Hinh 5.
6.- Hinh 6.
7.- Hinh 7.
8.- Hinh 8.
9.- Ket thuc chuong trinh
Chon chuc nang can thuc hien (0-9):

```

🔗 **Yêu cầu:** tổ chức chương trình thành các hàm chức năng, mỗi hàm vẽ 1 hình

49/. Thực hiện tương tự bài tập trước để tạo menu cho người dùng chọn in mỗi lần 1 hình trong các hình sau (hình 9-16). Giả sử với $n=5$, các hình mà chương trình cần vẽ có dạng như sau:

9 <pre> * * * * * * * * * * * * * * * * * * * * * </pre>	10 <pre> * * * * * * * * * * * * * * * </pre>	11 <pre> * * * * * * * * * * </pre>	12 <pre> * * * * * * * * * * * * * * * </pre>
13 <pre> * </pre>	14 <pre> * * * * * * * * * * </pre>	15 <pre> * </pre>	16 <pre> * * * * * * * * * * </pre>

50/. Cho nhập số nguyên dương n . Kiểm tra xem n có phải là số nguyên tố hay không?

51/. Cho nhập số nguyên dương n . Liệt kê các số nguyên tố $< n$.

52/. Cho nhập số nguyên dương n , liệt kê các ước số của n là số nguyên tố.

Ví dụ: Nhập $n=36$. Các ước số của 36 gồm 1, 2, 3, 4, 6, 9, 12, 18

Nhưng chỉ in ra: các số vừa là ước số của 36, vừa là số nguyên tố: 2, 3

53/. Cho nhập số nguyên dương n . Đếm các số nguyên tố $< n$.

54/. Cho nhập số nguyên dương n . Tìm số nguyên tố đầu tiên $< n$ và có giá trị gần với n nhất.

55/. Cho nhập số nguyên dương n . Tìm số nguyên tố đầu tiên $> n$ và có giá trị gần với n nhất.

56/. Lucky Number

- (Theo Wikipedia) *Lucky Number* (số may mắn) là số được định nghĩa theo quá trình sau: bắt đầu với số nguyên dương x và tính tổng bình phương y các chữ số của x , sau đó tiếp tục tính tổng bình phương các chữ số của y . Quá trình này lặp đi lặp lại cho đến khi thu được kết quả là 1 thì dừng (tổng bình phương các chữ số của số 1 chính là 1) hoặc quá trình sẽ kéo dài vô

tận. Số mà quá trình tính này kết thúc bằng 1 gọi là số may mắn. Số có quá trình tính kéo dài vô tận là *số không may mắn* hay còn gọi là **sad number** (*số đen đui*)

Ví dụ: 7 là số may mắn vì

$$7^2 = 49$$

$$4^2 + 9^2 = 97$$

$$9^2 + 7^2 = 130$$

$$1^2 + 3^2 + 0^2 = 10$$

$$1^2 + 0^2 = 1$$

- Minh họa những số may mắn dưới 500 là: 1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100, 103, 109, 129, 130, 133, 139, 167, 176, 188, 190, 192, 193, 203, 208, 219, 226, 230, 236, 239, 262, 263, 280, 291, 293, 301, 302, 310, 313, 319, 320, 326, 329, 331, 338, 356, 362, 365, 367, 368, 376, 379, 383, 386, 391, 392, 397, 404, 409, 440, 446, 464, 469, 478, 487, 490, 496.
- Yêu cầu cài đặt: Viết hàm liệt kê các *Lucky number* có giá trị ≤ 10000 .

Giả sử để xác định quá trình tìm *Lucky number* có kéo dài vô tận hay không, người ta cho lặp việc tìm này tối đa 100 lần. Do đó nếu quá trình lặp để tìm vượt trên 100 lần thì chương trình có thể kết luận số đó không là *Lucky number*.

- 57/. (*)¹ Giả sử số n được gọi là số may mắn là khi n chỉ chứa các ký số 6 hoặc 8. Ví dụ các số sau đây là số may mắn là 86, 6, 66, 868, 8, ...; còn các số sau đây là không may mắn: 99, 56, 87, ...

Yêu cầu: cho nhập số nguyên dương n . In ra số may mắn nhỏ hơn hoặc bằng và gần với n nhất.

Ví dụ: nhập $n = 686 \Rightarrow 686$

nhập $n = 87 \Rightarrow 86$

nhập $n = 1234 \Rightarrow 888$

- 58/. (*)² Viết chương trình cho nhập số nguyên dương n và nhỏ hơn 500 ($0 < n < 500$). Tìm số nguyên x sao cho X thỏa 2 điều kiện sau:

- X là bội số nhỏ nhất của n .
- X chỉ chứa các ký số 0 hoặc 9.

Ví dụ: Bội số của 7 là 9009

vì $9009/7=1287$

Bội số của 10 là 90

vì $90/10=9$

Bội số của 499 là 99900000099

vì $99900000099/499=200200401$

1.1.4. Khác

- 59/. Cho nhập 1 số nguyên dương có giá trị từ 0 đến 27. Tìm các bộ 3 số i, j, k sao cho $0 \leq i, j, k \leq 9$ và $i + j + k = n$.

☞ Gợi ý: để tạo tổ hợp 3 số có giá trị từ 0 đến 9, sử dụng phương thức `product` của `itertools` như sau:

```
itertools.product(range(10), range(10), range(10))
```

¹ Trích trong đề thi ACM/ICPC khu vực miền Nam năm 2018

² Trích trong đề thi dùng cho buổi thử máy trước khi thi *Olympic Tin Học toàn quốc năm 2018 tại Học Viện Bưu Chính Viễn Thông – Hà Đông - Hà Nội*

1.2. Strings

1.2.1. Thao tác với kiểu dữ liệu chuỗi

60/. Viết chương trình cho người dùng nhập 1 chuỗi (S).

- a. Lần lượt in ra chuỗi S dưới 3 dạng: tất cả là chữ hoa (in), tất cả là chữ thường và tất cả ký tự đầu của mỗi từ trong S là chữ hoa, các ký tự còn lại trong từ là chữ thường.

Ví dụ: S = 'HeLlO PyThON'

```
In ra UPPER: HELLO PYTHON
      lower: hello python
      Sentence case: Hello python
      Capitalize Each Word: Hello Python
      Swap UPPER to lower and lower to UPPER: hElLo pYtHoN
```

- b. Cho nhập tiếp 1 ký tự (ch). Đếm xem trong S có bao nhiêu ký tự ch. Yêu cầu thực hiện bằng 2 cách:

- Cách 1: sử dụng phối hợp lệnh *for* và *if*.
- Cách 2: sử dụng phương thức *count* của đối tượng chuỗi.

- c. Đếm xem trong S có bao nhiêu ký tự, bao nhiêu ký số.

- d. Đếm xem trong S có bao nhiêu ký tự in hoa, ký tự thường, ký số và ký tự loại khác.

- e. Đếm và in ra trong S có bao nhiêu nguyên âm (vowel - a, e, i, o, u) ký tự, bao nhiêu phụ âm (consonant - b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y, z). Lưu ý khi đếm không phân biệt chữ hoa, chữ thường.

Ví dụ: S='Sai Gon' sẽ in ra Có 3 nguyên âm: a i o

Có 3 phụ âm: S G n

☞ Gợi ý: dùng phương thức *count* của kiểu dữ liệu chuỗi để đếm.

61/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Xuất S dưới dạng: ký tự ở vị trí lẻ là ký tự thường, tại vị trí chẵn là ký tự in hoa. VD: ThÀnH PhỐ HỒ cHÍ mInH

62/. Viết chương trình cho người dùng nhập 1 chuỗi (S).

- a. Nếu chiều dài chuỗi lớn hơn 4, chương trình thực hiện đảo ngược thứ tự các ký tự trong S, ngược lại khi chiều dài chuỗi nhỏ hơn hay bằng 4, chương trình giữ nguyên chuỗi ban đầu. Ví dụ:

S='Python' ⇒ nohtyP

S='Data' ⇒ Data

- b. Gọi số lượng ký tự hoa trong S là *upp*, và gọi số lượng ký tự thường trong S là *low*. Nếu *upp* >= *low*, thực hiện đổi toàn bộ S thành chữ hoa, ngược lại, đổi toàn bộ S thành chữ thường.

Ví dụ: S='Python' ⇒ python

S='PythON' ⇒ PYTHON

63/. Viết chương trình cho người dùng nhập 1 chuỗi S và 1 từ (word). Yêu cầu:

- a. Đếm xem trong S có bao nhiêu từ *word* (khi đếm có phân biệt ký tự in hoa/ký tự thường).
- b. Đếm xem trong S có bao nhiêu từ *word* (khi đếm KHÔNG phân biệt ký tự in hoa/ký tự thường).

Ví dụ: với s='''Chiều chiều trước bến Văn Lâu

Ai ngồi, ai câu, ai sầu, ai thảm

Ai thương, ai cảm, ai nhớ, ai trông

Thuyền ai thấp thoáng ven sông

Đưa câu mái vẩy chạnh lòng nước non'''

Và với word='ai'. Kết quả sẽ in: số từ ai là 7

64/. Giả sử một chuỗi được gọi là hoàn chỉnh khi:

- Đầu và cuối chuỗi không chứa khoảng trắng (space).
- Giữa các từ chỉ cách nhau bởi 1 khoảng trắng.
- Dấu chấm và dấu phẩy phải đi liền với từ ngay trước mà không được cách bởi khoảng trắng.
- Nếu là bài thơ, các dòng phải được canh thẳng hàng (đều phải xuất phát từ đầu dòng).

Ví dụ: (ký tự `\` được diễn tả thay cho khoảng trắng)

Chuỗi chưa hoàn chỉnh	Chuỗi hoàn chỉnh
<code>Quê hương</code>	<code>Quê hương</code>
<code>Quê hương là chùm khế ngọt.</code>	<code>Quê hương là chùm khế ngọt.</code>
<code>Cho con trẻ hái mỗi ngày.</code>	<code>Cho con trẻ hái mỗi ngày.</code>
<code>Quê hương là đường đi học.</code>	<code>Quê hương là đường đi học.</code>
<code>Con về rợp bướm vàng bay.</code>	<code>Con về rợp bướm vàng bay.</code>
<code>Đố Trung Quân</code>	<code>Đố Trung Quân</code>

Viết chương trình cho người dùng nhập 1 chuỗi (S). Thực hiện xóa tất cả các khoảng trắng thừa.

65/. Viết chương trình cho người dùng nhập 1 chuỗi. Cho biết chuỗi đó có đối xứng (*palindrome*) hay không (không phân biệt ký tự hoa/thường)? Một số chuỗi đối xứng như: 'madam', 'abba', 'radar', 'Able was I ere I saw Elba'. Yêu cầu viết chương trình bằng 2 cách:

- Sử dụng toán tử index để đảo chuỗi
- Xử lý đảo từng ký tự trong chuỗi ban đầu X để có chuỗi Y, sau đó so sánh X và Y.

66/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Nhập thêm số lượng chuỗi cần tách (n) và ký tự (c) dùng làm căn cứ cho việc tách chuỗi. Thực hiện tách chuỗi S từ cuối về đầu thành n+1 đoạn dựa vào ký tự c.

Ví dụ: `S='Thành Phố Hồ Chí Minh', c=' '; n=3`
 \Rightarrow xuất ra ['Thành Phố', 'Hồ', 'Chí', 'Minh']

67/. Cho nhập số nguyên dương n. In ra các số nguyên dương X nhỏ hơn hay bằng n, sao cho X có chứa ít nhất một số 5 ở bất kỳ vị trí nào (hàng chục, hàng đơn vị, hàng trăm, ...). Các số trong kết quả được in ra cách nhau bởi dấu phẩy (,).

Ví dụ: nhập `n= 60`
 In ra 5, 15, 25, 35, 45, 50

68/. Cho người dùng nhập chuỗi (S) là 1 chuỗi số hệ nhị phân. Viết chương trình tìm độ dài lớn nhất chứa các số 0 liên tiếp trong S.

Ví dụ: `S= '1110000100000110' \Rightarrow` sẽ in ra 5

69/. Viết chương trình cho người dùng nhập 1 chuỗi S và tên của tag T (có trong HTML). Bao chuỗi S bên trong cặp tag T.

Ví dụ: `S='Python', T=i \Rightarrow` xuất ra `<i>Python</i>`
 Hay `S='Python', T=b \Rightarrow` xuất ra `Python`

70/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Tìm xem nếu từ 'poor' đi sau từ 'not' thì thay đoạn từ 'not' đến 'poor' thành duy nhất 1 từ 'good'. Các trường hợp khác sẽ giữ nguyên chuỗi (S).

Ví dụ: `S= 'The lyrics is not that poor!' \Rightarrow` xuất ra The lyrics is good!
`S= 'The lyrics is poor!' \Rightarrow` xuất ra The lyrics is poor!

71/. Cho nhập 1 chuỗi S. Chuyển tất cả các khoảng trắng nếu có ra phía đầu chuỗi S.

Ví dụ: (ký tự `\u` được diễn tả thay cho khoảng trắng)

S= "`uuuw3resource\u.ucomuu`" \Rightarrow "`uuuuuuw3resource.com`"

72/. Giả sử có định nghĩa về số đồng nhất như sau: số k được gọi là số đồng nhất khi $k > 0$ và các ký số có trong k đều giống nhau, ví dụ: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 111, 222, 1111, ...

Yêu cầu: cho nhập số nguyên n.

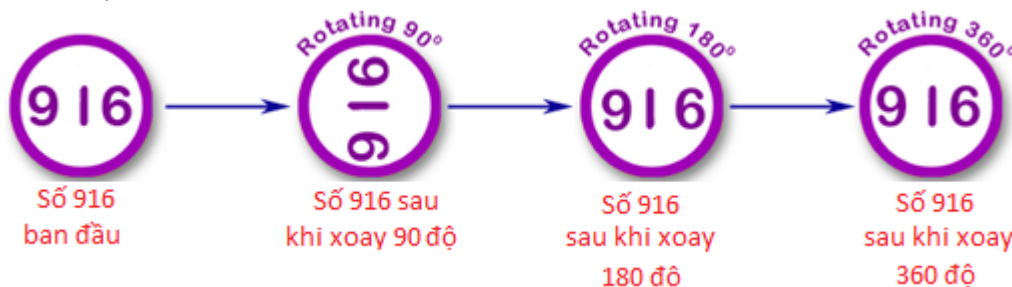
- Xét xem n có phải là số đồng nhất hay không?
 - Xử lý các số dưới dạng số nguyên.
 - Lần lượt chuyển các số sang dạng chuỗi để xét.
- Liệt kê các số đồng nhất nhỏ hơn hoặc bằng n bằng 2 cách:
 - Xử lý các số dưới dạng số nguyên.
 - Lần lượt chuyển các số sang dạng chuỗi để xét.

Số strobogrammatic

Số strobogrammatic

- Là một số có giá trị không đổi khi xoay số đó 180 độ (180°). Nhận xét: các số *strobogrammatic* chỉ chứa các số sau đây: 0, 1, 6, 8, 9. Trong đó các số 0, 1, 8 không bị thay đổi giá trị sau khi xoay, còn số 6 và 9 bị thay đổi (6 chuyển thành 9 và 9 chuyển thành 6).

Ví dụ 1:



Ví dụ 2: số 68910 sau khi xoay sẽ là 01689 \Rightarrow 68910 không phải là số

Strobogrammatic vì $68910 \neq 01689$

- Các số *strobogrammatic* đầu tiên là 0, 1, 8, 11, 69, 88, 96, 101, 111, 181, 609, 619, 689, 808, 818, 888, 906, 916, 986, 1001, 1111, 1691, 1881, 1961, 6009, 6119, 6699, 6889, 6969, 8008, 8118, 8698, 8888, 8968, 9006, 9116, 9696, 9886, 9966, ...
- Số nguyên tố strobogrammatic**
- Là số vừa là số nguyên tố, vừa là số *strobogrammatic*.
- Các số nguyên tố *strobogrammatic* đầu tiên là: 11, 101, 181, 619, 16091, 18181, 19861, 61819, 116911, 119611, 160091, 169691, 191161, 196961, 686989, 688889, ...
- Số strobogrammatic mở rộng**

Khi xoay các số dạng digital, ta thấy các số sau đây vẫn có ý nghĩa là 0, 1, 2, 5, 6, 8, 9. Trong đó các số 0, 1, 2, 5, 8 không bị thay đổi giá trị sau khi xoay, còn số 6 và 9 bị thay đổi (6 chuyển thành 9 và 9 chuyển thành 6)



Viết chương trình thực hiện các yêu cầu sau với kiểu dữ liệu cần xử lý là kiểu số nguyên:

- 73/. In ra các số strobogrammatic nhỏ hơn 1 triệu (1000000).
- 74/. In ra các số nguyên tố strobogrammatic nhỏ hơn 1 triệu (1000000).
- 75/. In ra các số strobogrammatic mở rộng nhỏ hơn 1 triệu (1000000).
- 76/. In ra các số nguyên tố strobogrammatic mở rộng nhỏ hơn 1 triệu (1000000).
- 77/. In ra các số nhỏ hơn 1 triệu (1000000) không phải là số strobogrammatic và không phải là số nguyên tố nhưng số strobogrammatic của số này lại là số nguyên tố.

Viết chương trình thực hiện lại các yêu cầu trên nhưng với kiểu dữ liệu cần xử lý là kiểu chuỗi:

- 78/. In ra các số strobogrammatic nhỏ hơn 1 triệu (1000000).
- 79/. In ra các số nguyên tố strobogrammatic nhỏ hơn 1 triệu (1000000).
- 80/. In ra các số strobogrammatic mở rộng nhỏ hơn 1 triệu (1000000).
- 81/. In ra các số nguyên tố strobogrammatic mở rộng nhỏ hơn 1 triệu (1000000).
- 82/. In ra các số nhỏ hơn 1 triệu (1000000) không phải là số strobogrammatic và không phải là số nguyên tố nhưng số strobogrammatic của số này lại là số nguyên tố.
- 83/. (**) Viết chương trình cho người dùng nhập 2 chuỗi (S1 và S2). Tìm đoạn văn bản ngắn nhất trong S1 sao cho đoạn này chứa tất cả các ký tự có trong S2. Nếu không có kết quả trả về chuỗi rỗng.

↳ Gợi ý: sử dụng Counter trong module collections
 Ví dụ: S1='abcdefgh'; S2='abcbcdgh' ⇒ 'abcd'

- 84/. (**) Viết chương trình cho người dùng nhập 2 chuỗi (S1 và S2). Tìm đoạn văn bản giống nhau và dài nhất có trong S1 và S2.

↳ Gợi ý: sử dụng SequenceMatcher trong module difflib
 Ví dụ: S1='abcdefgh'; S2='abcbcdgh' ⇒ 'abcd'

1.2.2. Thao tác với kiểu dữ liệu chuỗi dựa trên index

- 85/. Viết chương trình cho người dùng nhập họ tên (S). Thực hiện tách các phần họ, phần lót và phần tên. In kết quả ra màn hình.

Ví dụ: S='Nguyễn Thị Minh Khai'
 In ra: Họ: Nguyễn
 Lót: Thị Minh
 Tên: Khai

- 86/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Thực hiện hoán vị 2 ký tự đầu và cuối cho nhau.

Ví dụ: S='Python' ⇒ nythoP
 S='12345' ⇒ 52341

- 87/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Thực hiện xóa tất cả các ký tự tại vị trí có chỉ số là số lẻ.

Ví dụ: S='Python' ⇒ pto
 S='0123456789' ⇒ 02468

88/. Viết chương trình cho người dùng nhập 1 chuỗi S và số nguyên pos . Xóa ký tự tại vị trí pos của chuỗi (S). Nếu $Pos >$ chiều dài chuỗi S , xem như chuỗi S được giữ nguyên.

Ví dụ: $S='Python'$, $pos=0 \Rightarrow ython$
 $S='Python'$, $pos=3 \Rightarrow Pyton$
 $S='Python'$, $pos=5 \Rightarrow Pytho$

89/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Gọi ký tự đầu tiên của S là c . Tìm và thay thế tất cả các ký tự c có trong chuỗi thành ký tự '\$' (không thực hiện đối với ký tự đầu tiên của chuỗi S). Ví dụ: *restart* sẽ được đổi thành *resta\$t*, tương tự *madam* đổi thành *mada\$*

90/. Viết chương trình cho người dùng nhập 2 chuỗi S và R . Hoán vị 1 ký tự cuối của R và S cho nhau. Ví dụ: $S='abcd'$, $R='xyz'$ sẽ được đổi thành $S='abcz'$, $R='xyd'$

91/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Nếu chiều dài chuỗi S nhỏ hơn 3, giữ nguyên, ngược lại kiểm tra xem S kết thúc bằng "ing" hay không, nếu chưa có thì thêm ing vào S , ngược lại (khi đã có ing) thì thêm ly vào cuối (S).

Ví dụ: $S='ab' \Rightarrow$ xuất ra $'ab'$
 $S='str' \Rightarrow$ xuất ra $'string'$
 $S='string' \Rightarrow$ xuất ra $'stringly'$

92/. Viết chương trình cho người dùng nhập 1 chuỗi S và loại ngoặc định dùng (gồm 4 loại: [, {, (, <), trong đó, người dùng sử dụng ký tự đóng hay mở ngoặc đều được. Chèn chuỗi S vào giữa cặp ngoặc lồng nhau B.

Ví dụ: $S='Python'$, $B = '[' \Rightarrow$ xuất ra $'[[Python]]'$
 Hay $S='Python'$, $B = '}' \Rightarrow$ xuất ra $'{{Python}}'$
 Hay $S='Python'$, $B = '<' \Rightarrow$ xuất ra $'<<Python>>'$

93/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Xuất S dưới dạng: ký tự đầu và 3 ký tự cuối là chữ thường, còn lại là ký tự hoa. VD: THÀNH PHỐ HỒ CHÍ MINH

94/. Giả sử giữa 2 người A và B có quy ước về việc rút ngắn chuỗi ký tự trong văn bản rõ (*plain text*) bằng cách thay thế những ký tự liền kề và giống nhau như sau: ví dụ: *plaintext* chứa chuỗi ký tự 'YYYYY' sẽ được thay bằng '#5Y' trong bản mã (*cipher text*) hay 999.99 sẽ được thay bằng #39.#29.

Viết chương trình Python để khôi phục chuỗi gốc bằng cách nhập chuỗi nén (*cipher text*) với quy tắc này. Lưu ý ký tự # không được xuất hiện trong chuỗi ký tự được khôi phục (*plain text*).

Ví dụ: cipher text là **XY#6Z1#4023** sẽ xuất ra plain text là **XYZZZZZZ1000023**


Hay cipher text là **#39+1=1#30** sẽ xuất ra plain text là **999+1=1000**

95/. Viết chương trình cho người dùng nhập 1 chuỗi S chỉ chứa các ký tự từ a-z (ký tự thường). Cho biết S có chứa bao nhiêu ký tự phân biệt.

Ví dụ: $S='abc' \Rightarrow$ xuất ra 3
 Hay $S='abba' \Rightarrow$ xuất ra 2
 Hay $S='madam' \Rightarrow$ xuất ra 3

1.2.3. Xử lý chuỗi trên dữ liệu kiểu số

96/. Hãy kiểm tra các chữ của số nguyên dương n có giảm dần từ trái sang phải hay không?

 **Gợi ý:** chuyển số nguyên sang kiểu dữ liệu chuỗi sau đó duyệt chuỗi từ trái sang phải để xét chuỗi giảm dần.


97/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Cho biết chuỗi đó có là số nguyên (có thể âm hoặc dương) hay không?

Ví dụ: nhập `S = python3` \Rightarrow chuỗi 'không phải số nguyên'
`S = 1273` \Rightarrow chuỗi là số nguyên
`S = -1273` \Rightarrow chuỗi là số nguyên
`S = +1273` \Rightarrow chuỗi là số nguyên

 **Gợi ý:** sử dụng hàm `all()` để kiểm tra

98/. Viết chương trình nhập số thực x và số lượng số lẻ `dp(DecimalPlaces)`. Dùng phương thức `format` của chuỗi để in ra giá trị của x với `dp` số lẻ.

Ví dụ: $x = 3.1415926$ và `dp=4` \Rightarrow in ra $x = +3.1416$

 **Gợi ý:** để in ra số thực x với `dp` số lẻ và có dấu (âm hoặc dương), dùng định dạng sau:


```
sDP = '{ :+. ' + str(dp) + ' f } '
print(sDP.format(x))
```

99/. Viết chương trình cho người dùng nhập 1 chuỗi (S) bao gồm cả ký tự và ký số. Thực hiện tính tổng tất cả các ký số có trong S.

Ví dụ: `S='Python 3.7'` $\Rightarrow 10$

100/. Viết chương trình cho nhập 1 số nguyên dương n ($n > 0$). Chương trình thực hiện loại bỏ 1 số trong n sao cho giá trị các số còn lại trong n là nhỏ nhất.³

Ví dụ: `n = 21` \Rightarrow bỏ số 2 $\Rightarrow n = 1$
`n = 132` \Rightarrow bỏ số 3 $\Rightarrow n = 12$
`n = 104` \Rightarrow bỏ số 2 $\Rightarrow n = 4$
`n = 23198` \Rightarrow bỏ số 3 $\Rightarrow n = 2198$

 **Gợi ý:** nếu xét n dưới dạng chuỗi số thì khi `n[i] > n[i+1]` thì chữ số cần xóa là `n[i]`.

1.2.4. enumerate()

101/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Xuất ra vị trí của từng ký tự có trong chuỗi

Ví dụ: `S= "Sài Gòn"`, xuất ra Ký tự S xuất hiện tại vị trí 0
 Ký tự à xuất hiện tại vị trí 1
 Ký tự i xuất hiện tại vị trí 2
 Ký tự xuất hiện tại vị trí 3
 Ký tự G xuất hiện tại vị trí 4
 Ký tự ò xuất hiện tại vị trí 5
 Ký tự n xuất hiện tại vị trí 6

102/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Tìm ký tự đầu tiên xuất hiện nhiều hơn 1 lần

Ví dụ: `S= "Sài Gòn"`, xuất ra Ký tự S xuất hiện tại vị trí 0

³ Trích trong đề thi ICPC Asia Regionals 2019 Online Preliminary Round (<https://www.codechef.com/ICPCIN19>)

1.3. Bitwise Operator

103/ (*) Viết hàm thực hiện phép cộng giữa 2 số nguyên a và b nhưng không sử dụng phép cộng (+) mà chỉ sử dụng các phép toán trên bit.

☞ **Gợi ý**

```
Thực hiện lặp khi b!=0
data= a AND b
a = a XOR b
b= data SHIFT LEFT 1
return a
```

USER DEFINE FUNCTION - MODULE - PACKAGE

2.1. Xây dựng hàm

1/. Cho nhập 2 số a và b. Lần lượt thực hiện hoán vị 2 biến a, b bằng 2 cách sau:

- Không sử dụng hàm
- Có sử dụng hàm

2/. Viết chương trình cho nhập số nguyên n. In ra các ước số của n và số lượng ước số của n.

Ví dụ: với n=15, chương trình sẽ in ra

Các ước số của 15 là: [1, 3, 5, 15]

Số lượng ước số của 15 là: 4

3/. Viết chương trình chuyển đổi giữa 3 đơn vị đo lường cm, inch, foot bằng cách tạo menu như minh họa sau. Yêu cầu mỗi chuyển đổi sẽ được tổ chức thành 1 hàm riêng:

```
CHƯƠNG TRÌNH CHUYỂN ĐỔI GIỮA 3 ĐƠN VỊ cm, inch và foot
1- Từ cm sang inch
2- Từ cm sang foot
3- Từ inch sang cm
4- Từ inch sang foot
5- Từ foot sang cm
6- Từ foot sang inch
0- Kết thúc chương trình
Bạn chọn chức năng (0-6):
```

Biết rằng: centimeter (cm) = Inch / 2.54
 inch (in) = cm x 0.3937008
 foot (ft) = inches x 12

Sau khi hoàn tất, mở rộng thêm cho các đơn vị yard và mile

yard (yd) = feet x 3
 mile (mi) = yards x 1.760

4/. Viết hàm nhận tham số là 1 chuỗi (S). Nếu 2 ký tự đầu của s không phải là 'Is' thì thêm 'Is' vào trước s, ngược lại, nếu đã có thì trả về chuỗi s ban đầu.

5/. Viết hàm nhận tham số là 2 số nguyên. Nếu 1 trong 2 tham số không phải kiểu số nguyên thì sử dụng lệnh *raise* để định nghĩa bổ sung câu báo lỗi là "*Hàm chỉ nhận 2 tham số là số nguyên*" cho *TypeError Exception*.

6/. Cho người dùng nhập 1 (hoặc nhiều - tùy theo yêu cầu của từng bài) số nguyên. Kiểm tra nếu số nhập vào không đúng yêu cầu của đề bài thì cho nhập lại cho đến khi nhập đúng. Sau khi nhập đúng, hàm trả về 1 (hoặc nhiều) số đã nhập để chương trình thực hiện tiếp các yêu cầu còn lại của từng câu.

Yêu cầu: Mỗi bài tập sau sẽ được viết thành 2 hàm riêng biệt, hàm 1 kiểm tra việc nhập dữ liệu và trả về các số đã nhập; hàm 2 xử lý yêu cầu sau khi việc nhập thành công.

- a. Điều kiện: $n > 0$. Nếu nhập đúng, chương trình in ra các số chẵn nhỏ hơn n .
- b. Điều kiện: $0 \leq n \leq 9$. Nếu nhập đúng, chương trình in ra cách đọc của số vừa nhập.
- c. Điều kiện: $n < 10$ hoặc $n > 50$. Nếu nhập đúng, chương trình in ra n số ngẫu nhiên.
- d. Điều kiện: n bội số của 5. Nếu nhập đúng, chương trình in ra bảng cửu chương 5.

- e. Điều kiện: **n nằm trong các số (2, 4, 6, 8, 10)**. Nếu nhập đúng, chương trình cho nhập tiếp 2 số nguyên i, j . Hãy tính và in ra cấp số cộng với số hạng đầu là i , công sai là j và số lượng phần tử của dãy là n .
- 7/. Viết hàm nhận tham số là 1 số nguyên n . In ra chuỗi số Fibonacci gồm n số.
Ví dụ: với $n=9$, sẽ in ra 1 1 2 3 5 8 13 21 34
- 8/. Viết chương trình cho nhập 2 số a và b . Viết hàm tìm ước số chung lớn nhất (greatest common divisor - GCD) của 2 số a và b rồi in kết quả ra màn hình.
- 9/. **Số thân thiện**: Số tự nhiên có rất nhiều tính chất thú vị. Ví dụ với số 23, số đảo ngược của nó là 32. Hai số này có ước chung lớn nhất là 1. Những số như thế được gọi là số thân thiện, tức là số 23 được gọi là số thân thiện, số 32 cũng được gọi là số thân thiện.
- Hãy nhập vào 2 số nguyên a, b ($10 \leq a \leq b \leq 30000$). Hãy in ra các số thân thiện trong khoảng từ a đến b (kể cả a và b) và số lượng số thân thiện đã in ra.
- 10/. Viết chương trình tìm bội số chung nhỏ nhất (LCM - Least Common Multiple hay Lowest Common Multiple) của các số từ 1 đến 10. In ra LCM và thời gian thực hiện việc tìm LCM.
- Mở rộng: Thực hiện tương tự nhưng tìm LCM cho các số từ 1 đến 100.
- 11/. Viết chương trình lặp lại nhiều lần việc cho người dùng nhập 1 số nguyên n , xét xem số đó có phải là số nguyên tố hay không? Chương trình chỉ kết thúc khi người dùng nhập $n=-1$ hoặc nhấn tổ hợp phím Ctrl+C
- 12/. Nhập vào 3 số nguyên dương a, b, c . Kiểm tra xem 3 số đó có lập thành 3 cạnh của một tam giác hay không? Nếu có hãy cho biết:
- Chu vi, diện tích của tam giác
 - Chiều dài mỗi đường cao của tam giác.
 - Tam giác đó thuộc loại nào? (vuông cân, cân, vuông, đều, ...).
- **Nhắc lại:**
- Công thức tính diện tích $s = \sqrt{p(p-a)(p-b)(p-c)}$
 - Công thức tính các đường cao: $h_a = 2s/a, h_b = 2s/b, h_c = 2s/c$.
- (Với p là nửa chu vi của tam giác).
- **Yêu cầu:** tổ chức chương trình thành các hàm chức năng, mỗi hàm chỉ giải quyết 1 công việc nhất định. Ví dụ: hàm xét tam giác có hợp lệ hay không?, hàm xét tam giác đó có phải là tam giác đều hay không? ...

2.2. Anonymous function (hàm ẩn danh)

2.2.1. Xây dựng hàm ẩn danh với đối tượng number

- 13/. Viết chương trình Python sử dụng lambda để tính cho các trường hợp sau:
- Hàm nhận 1 đối số là số nguyên n và trả về trị tuyệt đối của n .
 - Hàm nhận 1 đối số là số nguyên n và trả về giá trị của $n+15$.
 - Hàm nhận 2 đối số là số nguyên (x, y) , trả về tích của x và y .
 - Hàm nhận 1 đối số là số nguyên n . Cho biết n có là bội số của 13 hoặc 19 hay không?
 - Hàm nhận 1 đối số là số thực r là bán kính của hình tròn. Cho biết diện tích, chu vi hình tròn.
 - Hàm nhận 2 đối số là số thực d, r là chiều dài và chiều rộng của hình chữ nhật. Cho biết chu vi và diện tích hình chữ nhật.
- **Gợi ý thực hiện câu e và f:** hàm lambda trả về nhiều giá trị.

- g) Hàm nhận 1 đối số là số nguyên n. Cho biết n có là số chính phương (*square number*) hay không? (số chính phương là số có căn bậc hai là 1 số nguyên như: 4, 9, 16, ...).
- h) Hàm nhận 3 tham số là số nguyên (a, b, c). Cho biết a, b, c có là 3 cạnh hợp lệ của 1 tam giác hay không?

Yêu cầu thực hiện của 2 câu g và h:

- Cách 1: hàm lambda chỉ trả về kết quả True hoặc False.
- Cách 2: sử dụng if...else để in ra thông báo kết quả ngay trong thân hàm lambda. Ví dụ in ra kết quả: "9 là số chính phương" hay "8 không phải số chính phương".

14/. Cho một hàm đã được định nghĩa như sau:

```
def lambda_in_function(n):
    return lambda x : x * n
```

Lần lượt gọi thực hiện nhân giá trị n=15 với x gồm các giá trị từ 2 đến 10

2.2.2. Xây dựng hàm ẩn danh với đối tượng string

15/. Hàm nhận 1 đối số là số nguyên n. Cho biết n có là số nguyên tố hay không? Viết hàm lambda bằng 2 cách: có và không có gọi hàm bên ngoài (User Define Function)

📖 Gợi ý: sử dụng kỹ thuật comprehension kết hợp với hàm all() khi hàm lambda không gọi hàm bên ngoài.

16/. Hàm nhận 3 tham số là số nguyên (a, b, c). Cho biết a, b, c có là 3 cạnh hợp lệ của 1 tam giác hay không? Nếu là 3 cạnh hợp lệ của tam giác, cho biết đó là tam giác gì? (thường, cân, đều, vuông, ...).

17/. Viết chương trình cho nhập 1 chuỗi S, lần lượt sử dụng lambda để xét các trường hợp sau:

- Chuỗi S có bắt đầu bằng ký tự 'P' (in hoa) hay không?
- Chuỗi S có bắt đầu bằng ký tự 'P' (in hoa) hoặc 'p' (in thường) hay không?

📖 Gợi ý: thực hiện câu a và b bằng cả 2 cách sau:

- Cách 1 - chuỗi sFind chỉ có duy nhất 1 ký tự: `strObject[0]== strFind`.
- Cách 2 - chuỗi sFind có thể là 1 hoặc nhiều ký tự: sử dụng phương thức `strObject.startswith(strFind)`. Phương thức này sẽ trả về kết quả là **True** hoặc **False**.

- Chuỗi S có bắt đầu bằng 1 nguyên âm hay không?
- Chuỗi S có phải là chuỗi *palindrome* hay không? Một chuỗi được gọi là *palindrome* khi các ký tự trong chuỗi đối xứng nhau.

Ví dụ: chuỗi sau là chuỗi *palindrome*: ABLE WAS I ERE I SAW ELBA

2.2.3. Xây dựng hàm ẩn danh với hàm do người dùng định nghĩa (User define function)

18/. Số Palindrome

📖 Số Palindrome (hoặc Palindromic)

- (Theo Wikipedia) Là một số vẫn giữ nguyên giá trị khi các chữ số của nó được đảo ngược. Hay nói cách khác là số đối xứng.
- 30 số thập phân palindrome đầu tiên là: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191, 202, ...

Số nguyên tố Palindrome

- (Theo Wikipedia) Là số nguyên tố viết xuôi hay viết ngược vẫn chỉ cho ra một số.
- Các số nguyên tố Palindrome dưới 20000 gồm: 2, 3, 5, 7, 11, 101, 131, 151, 181, 191, 313, 353, 373, 383, 727, 757, 787, 797, 919, 929, 10301, 10501, 10601, 11311, 11411, 12421, 12721, 12821, 13331, 13831, 13931, 14341, 14741, 15451, 15551, 16061, 16361, 16561, 16661, 17471, 17971, 18181, 18481, 19391, 19891, 19991.

Yêu cầu: Sử dụng lambda, viết chương trình

- In ra các số Palindrome từ 2 đến 1000000.
- In ra các số nguyên tố Palindrome từ 2 đến 100000.

2.2.4. *Xây dựng hàm ẩn danh với đối tượng datetime*

19/. Viết chương trình sử dụng lambda để lần lượt tách ngày, tháng, năm và giờ phút giây của thời điểm hiện tại

Ví dụ: thời điểm hiện tại là 2019-06-08 10:13:35.232478

Sẽ in ra màn hình: Năm hiện tại: 2019

Tháng hiện tại: 6

Ngày hiện tại: 8

Thời gian hiện tại: 10:13:35.232478

2.2.5. *Sử dụng các module khác để xây dựng hàm ẩn danh*

20/. Viết chương trình sử dụng lambda với đối số là n, với n là số lượng số đầu tiên trong dãy Fibonacci. Hàm trả về 1 list chứa n số trong dãy Fibonacci

 Gợi ý: sử dụng hàm reduce trong module functools

2.3. **Xây dựng hàm đệ quy (Recursion)**

21/. Cho nhập số nguyên n. Tính tổng các chữ số có trong n

Ví dụ: $n=345 \Rightarrow \text{tổng} = 12$

22/. Viết hàm tính giai thừa (factorial number) của một số nguyên dương (n).

Với $1! = 1$; $2! = 1 \times 2$; $3! = 1 \times 2 \times 3$; ... $n! = 1 \times 2 \times 3 \times \dots \times n$

23/. Viết hàm nhận tham số là 1 số nguyên dương (n). Tính số hạng thứ n của chuỗi Fibonacci. Ví dụ với $n=7$ sẽ in ra số 13, $n=8$ sẽ in ra 21, ...

↪ Nhắc lại: Chuỗi số Fibonacci: 1 1 2 3 5 8 13 ...

Dãy Fibonacci được Định nghĩa truy hồi như sau:

- $F_0 = F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ nếu $n \geq 2$

24/. Viết hàm nhận 2 tham số là 2 số nguyên dương a, b.

- Tính a^b . Ví dụ: $a=2, b=3 \Rightarrow 2^3=8$
- Tìm ước số chung lớn nhất (greatest common divisor -gcd) của a và b.

Ví dụ: với $a=2$ và $b=7 \Rightarrow \text{gcd}(a,b)=1$

với $a=6$ và $b=18 \Rightarrow \text{gcd}(a,b)=6$

25/. Viết hàm nhận tham số là 1 số nguyên dương (n). Tính tổng của S trong các trường hợp sau:

a. $S = n + (n-2) + (n-4) + \dots$ cho đến khi $(n-x) \leq 0$.

Ví dụ: $n=6 \Rightarrow \text{tổng} = 6 + 4 + 2 = 12$

$n=7 \Rightarrow \text{tổng} = 7 + 5 + 3 + 1 = 16$

b.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

c.

$$S = 1 + \frac{1}{2^1} + \frac{1}{2^2} + \dots + \frac{1}{2^n}$$

Ví dụ: với $n=2$ sẽ in ra $S=1.75$

với $n = 3$ sẽ in ra $S = 1.875$

26/. Viết hàm nhận tham số là 1 list.

a. Tính tổng các giá trị có trong List trong trường hợp các phần tử trong List là phân tử đơn.

Ví dụ: [2, 4, 5, 6, 7]

b. Tính tổng các giá trị có trong List trong trường hợp các phần tử trong List có thể là 1 phần tử đơn hoặc 1 subList. Ví dụ: [1, 2, [3, 4], [5, 6]]

c. In các giá trị có trong List theo chiều từ đầu đến cuối list

d. In các giá trị có trong List theo chiều từ cuối về đầu list

27/. Viết hàm nhận 2 tham số là số nguyên, với tham số thứ nhất số nguyên hệ thập phân (n) và tham số thứ hai (base) cơ số cần đổi (2,8,16). Thực hiện đổi n sang cơ số base. Kết quả trả về của hàm là chuỗi kết quả.

Ví dụ: $n=179, \text{ base}=2 \Rightarrow 179 \text{ Dec} = 10110011 \text{ Binary}$

$n=179, \text{ base}=8 \Rightarrow 179 \text{ Dec} = 263 \text{ Octal}$

$n=179, \text{ base}=16 \Rightarrow 179 \text{ Dec} = \text{B3 Hexa Decimal}$

28/. Giả sử chuỗi S được gọi là chuỗi tuần tự 4 khi chuỗi được bắt đầu bởi 4 số 1, giá trị từ số thứ 5 trở đi sẽ bằng tổng của 4 số liền trước đó. Ví dụ: $S = 11114713254994 \dots$ Viết chương trình cho nhập số nguyên dương n. Hãy in ra số thứ n của dãy chuỗi (S).

Ví dụ: $n=5$ in ra **4**

$n=6$ in ra **7**

$n=7$ in ra **13**

$n=8$ in ra **25**

29/. Biết lãi suất vay được tính theo lũy tiến (lãi tháng trước được cộng dồn vào vốn vay) và lãi suất 1 tháng là 5%. Cho người dùng nhập số tiền (X) và số tháng cần vay (m). Cho biết số tiền khách hàng phải trả sau thời gian m tháng. Yêu cầu: nếu số tiền vay dưới 1 triệu thì số tiền kết quả sẽ làm tròn đến phần trăm, ngược lại sẽ làm tròn số tiền đến phần ngàn.

2.4. Datetime module

30/. Viết chương trình sử dụng phương thức strftime của đối tượng thuộc class date hoặc datetime để in ra các thông tin như sau:

Thông tin cần hiển thị	Kết quả hiển thị (ví dụ ngày hiện tại là 18/9/2019)
Ngày giờ hiện tại	2019-09-18 19:25:58.737442
Năm hiện tại	2019
Tháng hiện tại bằng chữ	September
Tuần hiện tại là tuần thứ mấy trong năm	37
Tuần hiện tại là tuần thứ mấy trong tháng	3
Ngày hiện tại là ngày thứ mấy trong năm	261
Ngày dương lịch hiện tại là ngày	18
Thứ của ngày hiện tại	Wednesday
Giờ phút giây hiện tại	19:25:58

31/. Cho nhập ngày, tháng, năm của 2 ngày. Cho biết số ngày cách nhau giữa 2 ngày.

32/. Cho nhập 1 chuỗi ngày (S) theo dạng '**Sep 18 2019 2:43PM**'. Chuyển chuỗi S sang kiểu ngày.

33/. Lấy giá trị khởi đầu (0 giờ, 0 phút, 0 giây) của ngày hiện tại. VD: 2019-09-18 00:00:00

34/. Sử dụng `datetime.timedelta` để thêm 5 giây vào thời gian hiện tại

35/. Cho nhập tháng (m) và năm dương lịch (y), yêu cầu thực hiện:

- Cho biết năm y có là năm nhuận hay không?
- Tháng m của năm y có bao nhiêu ngày?
- Ngày cuối cùng của tháng m là thứ mấy?
- In ra lịch của tháng và năm tương ứng. Lưu ý cần kiểm tra giá trị của tháng phải nằm trong khoảng từ 1 đến 12.

```
Input the year : >? 2019
Input the month : >? 8
          August 2019
Mo Tu We Th Fr Sa Su
                1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

36/. Viết chương trình cho nhập năm dương lịch X (ví dụ X=2020).

Chương trình in ra tất cả các ngày chủ nhật có trong năm X, với ngày được xuất ra có dạng yyyy/mm/dd.

☞ **Gợi ý:**

- Sử dụng hàm `date` để lấy ngày 1 tháng 1 của năm X rồi gán vào đối tượng kiểu ngày (`mydate`)
- Sử dụng phương thức `weekday` của đối tượng ngày để lấy số thứ tự trong tuần của đối tượng `mydate`.
- Sử dụng hàm `timedelta` để tính giá trị ngày chủ nhật đầu tiên và cho các ngày chủ nhật kế tiếp.

CÁC ĐỐI TƯỢNG DẠNG DANH SÁCH TRONG PYTHON

(Iterator object or Sequences types)

3.1. Lists

3.1.1. List

1/. Cho trước 1 list như sau:

```
datalist = [1452, 11.23, 1 + 2j, True, 'w3resource',
            (0, -1), [5, 12], {"class": 'V', "section": 'A'}]
```

Viết chương trình cho biết kiểu dữ liệu của từng thành phần trong list.

Gợi ý: dùng hàm `type(object)` để biết kiểu dữ liệu của object. Các kiểu cần xét trong bài tập này là `int`, `float`, `complex`, `bool`, `list`, `tuple`, `dict`, `str`. Nếu kiểu của dữ liệu không thuộc các loại vừa liệt kê, chương trình in ra `Unknown`

3.1.2. Number List

2/. Cho nhập 1 dãy số cách nhau bởi dấu phẩy (S).

a. Chương trình phát sinh ra 1 list từ chuỗi số trên. Ví dụ: `S = '1, 3, 9, 2, 8, 4, 7'`
 \Rightarrow `list=[1, 3, 9, 2, 8, 4, 7]`

b. Tính tổng các số trong S.

c. Tìm số nhỏ nhất trong S.

3/. Viết chương trình lần lượt thực hiện các yêu cầu sau:

a. Tạo 1 danh sách (list) chứa nhiều số nguyên có giá trị tùy ý, ví dụ:

```
lst = [1, -1, 2, 0, 5, 8, -13, 21, -34, 55, 87, 0]
```

b. Viết chương trình in tất cả các phần tử (elements) trong list có giá trị nhỏ hơn 5 ra màn hình.

c. Thay vì in trực tiếp các phần tử ra màn hình, chương trình tạo ra 1 list mới chứa các phần tử có giá trị nhỏ hơn 5. Sau đó in tất cả các số có trong list vừa tạo ra màn hình.

d. Cho người dùng nhập số m. Chương trình in ra các phần tử trong list có giá trị là bội số của m.

e. Viết hàm tính tỷ lệ phần trăm của 3 giá trị: số dương, zero (0) và số âm có trong list. Hàm trả về 3 tỷ lệ vừa tính. Dựa vào đó, chương trình chính in ra kết quả.

4/. Lần lượt cho nhập 3 giá trị: chuỗi, số nguyên, số thực vào 3 biến a, b, c. Lần lượt đưa giá trị của 3 biến vào list L. Sau đó sử dụng index để đưa 3 giá trị đang có trong list L vào 3 biến x, y, z bằng 2 cách sau

- Cách 1: dùng 3 dòng lệnh

- Cách 2: dùng 1 dòng lệnh

5/. Viết chương trình cho người dùng nhập 1 số nguyên dương n. Tạo 1 list chứa tất cả các số nguyên i sao cho $i \leq n$ và n chia hết cho i. In list kết quả ra màn hình.

6/. Thực hiện mỗi yêu cầu yêu cầu sau đây thành một hàm chức năng. Viết chương trình gọi thực hiện các hàm này:

a. Cho nhập 1 số nguyên dương n. Cần kiểm tra nếu n nhập vào không phải là kiểu số hoặc giá trị của $n \leq 0$, chương trình sẽ yêu cầu nhập lại cho đến khi nhập đúng. Hàm không có tham số đầu vào, sau khi nhập thành công, hàm trả về số nguyên n

Gợi ý: sử dụng `try ... except` để kiểm tra kiểu dữ liệu của n.

- b. Viết hàm nhận tham số là n. Hàm thực hiện tạo list L gồm n phần tử với giá trị ngẫu nhiên nguyên dương từ 0 đến 1000.
- c. Viết hàm nhận tham số là list L, hàm thực hiện in ra các số chẵn có trong list L. Hàm kết thúc khi đã xét hết các giá trị có trong list L hoặc ngay sau khi in giá trị là 99 (giá trị 99 này có thể có trong list khi phát sinh ngẫu nhiên)
- d. Viết hàm nhận tham số là list L, hàm thực hiện in ra các số may mắn có trong list L. Giả sử số may mắn là số chỉ chứa các ký số 6 hoặc 8. Ví dụ số may mắn là 86, 6, 66, 868, 8, ...; các số sau đây là không may mắn: 99, 56, 87, ...
- e. Viết hàm nhận tham số là list L, hàm thực hiện xóa các số nguyên tố có trong L. Sau khi xóa xong, hàm trả về list kết quả
- 7/. Thực hiện mỗi yêu cầu yêu cầu sau đây thành một hàm chức năng, sau đó viết chương trình chính gọi thực hiện các hàm này:
- Cho nhập 1 số nguyên dương n có giá trị trong khoảng từ 50 đến 100. Nếu nhập sai, yêu cầu nhập lại, ngược lại khi nhập đúng hàm trả về (return) n cho nơi gọi hàm.
 - Hàm nhận tham số là số nguyên n. Hàm thực hiện tạo list L gồm n phần tử với giá trị được phát sinh ngẫu nhiên trong khoảng từ 0 đến $2*n$.
 - Hàm nhận tham số là list L. In ra các số nguyên tố có trong L.
 - Hàm nhận tham số là list L. In ra các số chính phương có trong L.
 - Hàm nhận tham số là list L. In ra các may mắn có trong L.
 - Hàm nhận tham số là list L. Hàm thực hiện xóa các số nguyên tố có trong L.
 - Hàm nhận tham số là list L. Hàm thực hiện xóa các số chính phương có trong L.
- 📖 **Nhắc lại**
- Số chính phương (square number hay perfect square)
 - Là số có căn bậc hai là một số nguyên.
 - Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3.
 - Số nguyên tố là số tự nhiên lớn hơn 1 và chỉ có đúng hai ước số là 1 và chính nó.
 - Số may mắn: là số chỉ chứa các ký số 6 hoặc 8.
- 8/. Cho nhập 1 số nguyên n, tạo list L gồm n phần tử với giá trị ngẫu nhiên từ 0 đến 100. Viết hàm trả về chuỗi số kết nối từ các số có trong list L. Lưu ý chuỗi kết quả có kèm dấu ngăn cách phần ngàn như minh họa sau:
- $$L = [52, 99, 3, 60, 93, 97] \Rightarrow 52.993.609.397$$
- 9/. Cho nhập n. Xét xem n có phải là số nguyên tố hay không?
- Yêu cầu: sử dụng list chứa các ước số của n và nhỏ hơn n (không gồm số 1). Sau đó xét nếu list rỗng thì n là số nguyên tố, ngược lại n không là số nguyên tố. Lần lượt thực hiện tạo list bằng 2 cách: cách thông thường và List Comprehensions.
- 10/. Viết chương trình gồm các hàm (function) để thực hiện các chức năng như sau:
- Cho người dùng nhập nhiều lần các số nguyên dương. Việc nhập sẽ kết thúc khi người dùng nhập số âm. Đưa tất cả các số đã nhập (không kể số âm nhập cuối cùng) vào list L. Hàm trả về list L.
 - Hàm trả về tổng các số có trong List.

📖 Gợi ý: sử dụng hàm sum(listName) để tính tổng các số có trong list.
 - Hàm nhận tham số là list L và số nguyên X. Tìm xem x có trong list chứa các số vừa nhập hay không? Nếu có, cho biết x xuất hiện bao nhiêu lần?

📖 Gợi ý: sử dụng phương thức count của đối tượng list (listName.count(x)).
 - Hàm nhận tham số là list L và số nguyên x. Cho biết X có lớn hơn tất cả các số có trong list hay không? Nếu không, hãy in ra các số có trong list và lớn hơn x.

 **Gợi ý:** sử dụng hàm `max(listName)` để tìm số lớn nhất trong `listName`.

e. Hàm trả về tổng khoảng cách giữa tất cả các cặp số có trong L.

Ví dụ: `list L= [1, 2, 3]` in ra 'Tổng khoảng cách giữa các số là: 4'

Giải thích: vì $|1-2| + |2-3| + |1-3| = 1 + 1 + 2 = 4$

11/. Thực hiện mỗi yêu cầu yêu cầu sau đây thành một hàm chức năng. Viết chương trình gọi thực hiện các hàm này:

a. Viết hàm không nhận tham số đầu vào, hàm cho người dùng nhập nhiều lần các số nguyên dương. Chương trình cần kiểm tra số do người dùng nhập vào có thể là số âm hoặc số dương nhưng phải là số nguyên. Sau mỗi lần nhập, chương trình sẽ hỏi người dùng có muốn nhập nữa hay không (Yes/No). Nếu người dùng nhấn phím bất kỳ, chương trình cho người dùng nhập tiếp. Ngược lại nếu chọn No ('N' hoặc 'n'), hàm trả về list L chứa các số đã nhập.

b. Viết hàm nhận tham số là list L, hàm xóa các số chính phương có trong list L.

 **Số chính phương** (*square number hay perfect square*)

- Là số có căn bậc hai là một số nguyên.
- Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3.

c. Viết hàm nhận tham số là list L, hàm tính và trả về trung bình cộng các số âm, trung bình các số dương.


d. Viết hàm nhận tham số là list L, hàm trả True nếu các số trong list đã được sắp xếp tăng dần, ngược lại trả về False.

Biết rằng dãy số được gọi là tăng dần khi số đứng trước luôn lớn hoặc bằng số đi sau.

12/. Viết chương trình cho người dùng nhập 1 số nguyên dương n. sắp xếp các số trong n thành từ nhỏ đến lớn (nMin) và từ lớn đến nhỏ (nMax). Tìm hiệu của nMax và nMin.

Ví dụ nhập `n=1423`

\Rightarrow `nMin=1234, nMax=4321, nMax - nMin = 4321-1234 = 3087`

 **Gợi ý:** Đưa từng số trong n vào list. Sử dụng các phương thức join và sort của dữ liệu chuỗi để tìm nMin và nMax, từ đó suy ra kết quả hiệu

13/. Tạo ngẫu nhiên số nguyên n đại diện cho số lượng phần tử sẽ có trong listA (với $5 \leq n \leq 20$).

- Tạo listA gồm n phần tử với giá trị ngẫu nhiên từ 1 đến 100.
- Tạo listB bằng cách chỉ chọn những số chẵn có trong listA.
- In 2 list ra màn hình.

14/. Viết chương trình tạo 2 danh sách (A, B) với giá trị ngẫu nhiên (có thể trùng nhau) trong khoảng 1-X bằng cách cho người dùng nhập:

- Giá trị của X.
- Số lượng phần tử có trong từng list nA, nB.
- Range của số ngẫu nhiên sẽ được tạo (Giá trị nhỏ nhất và giá trị lớn nhất)

a. Tạo danh sách Result chứa những số có trong cả A và B.

b. **Mở rộng:** loại đi những số trùng nhau (nếu có) trong Result.

- Lần lượt duyệt từng phần tử X trong list L để tạo một (hoặc nhiều) newList sẽ chứa X và các hoán vị sẽ có với các List đã có trong đó.

Minh họa: `L=[1, 2, 3]`

- o `resultList=[[]]`
- o Xét số đầu tiên (số 1)
- o Tạo 1 hoặc nhiều newList bằng cách kết hợp số 1 với các sublist đang có trong resultList \Rightarrow `resultList=[[1]]`
- o Xét số kế tiếp (số 2)
- o Tạo 1 hoặc nhiều newList bằng cách kết hợp số 2 với các sublist đang có trong resultList bằng cách chèn số 2 vào các vị trí từ 0 đến `len(subList)` \Rightarrow `resultList=[[2,1], [1,2]]`
- o Xét số kế tiếp (số 3)
- o Tạo 1 hoặc nhiều newList bằng cách kết hợp số 3 với các sublist đang có trong resultList bằng cách chèn số 3 vào các vị trí từ 0 đến `len(subList)` \Rightarrow
 - Chèn số 3 vào sublist `[2,1]` (đang có trong resultList) sẽ phát sinh ra được 3 sublist khác là `[3,2,1]`, `[2,3,1]`, `[2,1,3]`.
 - Tương tự, chèn số 3 vào sublist `[1,2]` (đang có trong resultList) sẽ phát sinh ra được thêm 3 sublist nữa là `[3,1,2]`, `[1,3,2]`, `[1,2,3]`.

- 19/. Tìm các số nguyên tố nhỏ hơn 1000 bằng giải thuật sàng Erastosthene (giải thuật sàng Erastosthene dùng phương pháp đánh dấu để loại bỏ những số không phải là số nguyên tố. Giải thuật có từ một nhận xét rằng nếu k là số nguyên tố thì các số $2*k, 3*k, \dots, n*k$ sẽ không là số nguyên tố (vì đã vi phạm định nghĩa về số nguyên tố).

Yêu cầu: sử dụng list để tìm ra các số nguyên tố <1000 dựa trên giải thuật Erastosthene.

- 20/. (*)Viết chương trình minh họa cho những số <100 về giả thuyết của nhà toán học người Đức - Christian Goldbach (1690-1764, trong một bức thư ngày 07/6/1742 gửi cho nhà toán học Leonhard Euler): **mọi số tự nhiên chẵn lớn hơn 2 đều có thể viết được thành tổng của hai số nguyên tố** (giả thuyết này đã được chỉ ra là đúng tới 4×10^{18}).

Ví dụ: $4 = 2 + 2$, $8 = 5 + 3$, $20 = 13 + 7$

- 21/. Viết chương trình cho người dùng thực hiện lặp đi lặp lại nhiều lần việc nhập 2 số nguyên sl (số lượng) và tong(tổng). Chương trình tạo ra tổ hợp các số từ 0 đến 9 gồm sl phần tử và tổng các phần tử trong tổ hợp bằng với giá trị tong vừa nhập. In các tổ hợp thỏa điều kiện và số lượng tổ hợp thỏa điều kiện. Chương trình kết thúc khi người dùng nhập sl=0 và tong=0.

Ví dụ nhập sl=2 và tong=7. Kết quả sẽ in ra

Các tổ hợp gồm 2 phần tử có tổng = 7 là:

(0, 7) (1, 6) (2, 5) (3, 4)

Có 4 tổ hợp thỏa điều kiện

➤ Gợi ý: sử dụng `itertools.combinations` để tạo ra các tổ hợp

3.1.3. String List

- 22/. Tạo trước 1 list chứa tên gọi của các con thú (ví dụ: `['ant', 'bear', 'cat', 'dog', 'elephant', 'fish', 'goat', 'hippo']`). Thực hiện:
- Số lượng thú có trong danh sách và danh sách các con thú.
 - Cho người dùng nhập tên con thú cần tìm, in ra kết quả tìm kiếm: có hay không có con thú cần tìm.

- 23/. Viết chương trình cho người dùng nhập 1 chuỗi (S) và ký tự dùng để phân chia chuỗi (c). Thực hiện phân chia chuỗi S thành các phần sao cho các phần đó không còn chứa các ký tự

dùng để phân chia c. Cho biết số lượng chuỗi sau khi phân tách (không kể những chuỗi rỗng "").

Yêu cầu thực hiện bằng 2 cách:

- Cách 1: thực hiện đếm khi không xóa chuỗi rỗng ra khỏi list.
- Cách 2: thực hiện đếm sau khi đã xóa chuỗi rỗng ra khỏi list.

Ví dụ: với `s='https://www.w3resource.com/python-exercises/string'` và `c='/'`

Kết quả sẽ in ra:

Cách 1 - list gồm: `['https:', '', 'www.w3source.com', 'python-exercises', 'string']`

Có 4 chuỗi sau khi phân tách (không kể chuỗi rỗng)

Cách 2 - xóa chuỗi rỗng có trong list, list còn: `['https:', 'www.w3source.com', 'python-exercises', 'string']`

Có 4 chuỗi sau khi phân tách (không kể chuỗi rỗng)

24/. Viết chương trình tạo cho nhập 1 chuỗi S có chứa 2 từ “Python và “Java”. Chương trình thực hiện hoán đổi 2 từ này cho nhau.

Ví dụ: `S='Python is popular than Java'`

Kết quả in ra: `'Java is popular than Python'`

☞ Gợi ý:

- Tách các từ trong S vào 1 list.
- Duyệt từ đầu đến cuối list để thay thế từ.

☞ Mở rộng: Cho người dùng nhập chuỗi S và 2 từ tùy ý cần hoán đổi vị trí cho nhau.

25/. Cho nhập chuỗi (S). Đếm xem trong s có bao nhiêu ký tự là nguyên âm (AEIOU) mỗi loại. Lưu ý không phân biệt chữ hoa/thường.

☞ Gợi ý: sử dụng 2 list, list 1 lưu trữ các nguyên âm, list 2 lưu trữ các giá trị đếm được của nguyên âm có vị trí tương ứng với vị trí của giá trị đếm.

Ví dụ: `S=' Fundamental of Python'`, sẽ in ra: nguyên âm A có 2 ký tự

nguyên âm E có 1 ký tự

nguyên âm I có 0 ký tự

nguyên âm O có 2 ký tự

nguyên âm U có 1 ký tự

26/. Viết chương trình cho người dùng nhập 1 chuỗi S. Cho biết từ dài nhất và từ ngắn nhất trong S. Ví dụ:

`S='Write a Java program to sort an array of given integers using Quick sort Algorithm.'`

⇒ từ dài nhất: Algorithm.

⇒ từ ngắn nhất: a

27/. Cho nhập chuỗi (S). In và in ra tất cả các từ cùng có chiều dài lớn nhất có trong chuỗi.

Ví dụ: `S='PHP, Exercises, Backend, Databases'`

Sẽ in ra: Exercises Databases

28/. Viết chương trình nhập vào năm. In ra tên của năm âm lịch tương ứng.

Ví dụ: nhập năm 2019 in ra *Kỷ Hợi*.

Biết rằng:

CAN	Giáp	At	Bính	Đinh	Mậu	Kỷ	Canh	Tân	Nhâm	Quý
CHI	Tý	Sửu	Dần	Mão	Thìn	Tỵ	Ngọ	Mùi	Thân	Dậu Tuất Hợi

29/. Trong mật mã học, mật mã Caesar (*Caesar Cipher*, còn được gọi là mật mã dịch chuyển - *Shift Cipher*, mã của Caesar hoặc dịch chuyển Caesar), là một trong những kỹ thuật mã hóa (*Encryption*) đơn giản và được biết đến rộng rãi nhất. Nó là một loại mật mã thay thế, trong

đó mỗi chữ cái trong bản rõ (plain text) được thay thế bằng một chữ cái một số vị trí cố định trong bảng chữ cái dựa trên khóa k (key). Ví dụ: với dịch chuyển trái 3, D sẽ được thay thế bằng A, E sẽ trở thành B, v.v. Phương pháp này được đặt theo tên của Julius Caesar, người đã sử dụng nó trong thư tín riêng của mình.

Thông điệp được mã hóa bằng cách dịch chuyển xoay vòng từng ký tự đi k vị trí trong bảng chữ cái. Trường hợp với $k=3$ gọi là phương pháp mã hóa Caesar.

☞ **Minh họa:** Cho $plain\ text = x_1, x_2, x_3, x_4, x_5, x_6 = 'ATTACK'$, $k=17$, chiều dịch chuyển của ký tự là từ trái sang phải và bảng chữ cái các ký tự được mã hóa như hình sau (như vậy, nếu các ký tự ngoài những ký tự này sẽ giữ nguyên mà không thực hiện mã hóa:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

– Thực hiện mã hóa:

$$y_1 = x_1 + k \bmod 26 = 0 + 17 \bmod 26 = 17 \Rightarrow R$$

$$y_2 = 19 + 17 \bmod 26 = 10 \Rightarrow K$$

$$y_3 = 10 \Rightarrow K$$

$$y_4 = 17 \Rightarrow R$$

$$y_5 = 2 + 17 \bmod 26 = 19 \Rightarrow T$$

$$y_6 = 10 + 17 \bmod 26 = 1 \Rightarrow B$$

Bản mã hoàn chỉnh (Cipher text) $Y = y_1; y_2; y_3; y_4; y_5; y_6 = RKKRTB$

– Thực hiện giải mã: với $k=17$ và cipher text ($Y = y_1; y_2; \dots; y_6 = RKKRTB$)

$$x_1 = y_1 - k \bmod 26 = 17 - 17 \bmod 26 = 0 \Rightarrow A$$

$$x_2 = 10 - 17 \bmod 26 = -7 \bmod 26 = 19 \Rightarrow T$$

$$x_3 = 19 \Rightarrow T$$

$$x_4 = 0 \Rightarrow A$$

$$x_5 = 19 - 17 \bmod 26 = 2 \Rightarrow C$$

$$x_6 = 1 - 17 \bmod 26 = -16 \bmod 26 = 10 \Rightarrow K$$

Bản giải mã hoàn chỉnh (bản gốc - plan text) $X = x_1; x_2; \dots; x_6 = ATTACK$

☞ **Yêu cầu:** viết chương trình tạo menu với các chức năng sau, trong đó khi thực hiện mã hóa hoặc giải mã, chương trình sẽ yêu cầu nhập khóa k . Lưu ý: xem như văn bản chỉ gồm các ký tự chữ hoa.

- Nhập chuỗi
- Xuất chuỗi
- Mã hóa (*Encryption*)
- Giải mã (*Decryption*)

☞ **Mở rộng:**

- Văn bản bao gồm cả ký tự hoa và thường.
- Văn bản bao gồm tất cả các ký tự trong bộ mã ASCII nhưng bỏ qua các mã điều khiển (từ 0-30).
- Cho người dùng chọn chiều dịch chuyển (trái hoặc phải).

30%. Viết chương trình cho nhập 1 chuỗi (S). In ra ký tự xuất hiện nhiều nhất trong S. Lưu ý: có thể có nhiều ký tự cùng đạt nhiều nhất. Yêu cầu:

a. Chỉ in ra 1 ký tự đại diện cho những ký tự có số lần xuất hiện nhiều nhất.

Ví dụ: `S='madam' ⇒ in ra 'm'`

b. In ra tất cả các ký tự có số lần xuất hiện là nhiều nhất.

Ví dụ: `S = 'madam' ⇒ in ra 'ma' hoặc 'am'`

3.1.4. List & Two array dimension

31/. Viết chương trình cho nhập 2 số nguyên m (dòng) và n (cột) của một mảng hai chiều. Chương trình tạo giá trị cho các phần tử trong mảng 2 chiều A theo quy ước $A[i][j] = i * j$. Với dòng và cột được tính bắt đầu từ 0.

32/. Cần tạo 1 ma trận vuông cạnh n với giá trị các phần tử (ma trận sẽ có $n * n$ phần tử) do người dùng nhập vào.

- Đầu tiên, cho người dùng nhập 1 số nguyên dương n.
- Tiếp tục cho người dùng nhập $n*n$ số nguyên cho các phần tử.
- Yêu cầu: tính tổng dòng, tổng cột và in ra màn hình ma trận ban đầu, ma trận sau khi thêm tổng dòng, tổng cột.
- Minh họa: với $n=3$

1	2	3
5	6	7
9	10	11

Ma trận ban đầu

1	2	3	6
5	6	7	18
9	10	11	30
15	18	21	

Ma trận kết quả

- Mở rộng 1: giá trị của $n*n$ phần tử được phát sinh ngẫu nhiên trong khoảng từ -10 đến +10.
- Mở rộng 2: ma trận không vuông với m dòng và n cột.

33/. (*) Cho người dùng nhập 2 số nguyên dương n và m. Tạo ra 1 ma trận $n \times m$ chứa một trong 2 giá trị 0 hoặc -1. Một nhóm các ô chứa giá trị -1 sẽ cùng thuộc một miền liên thông khi các ô trên hoặc dưới hoặc phải hoặc trái của ô đang xét cũng chứa giá trị -1. Viết chương trình xác định số lượng miền liên thông có trong mảng 2 chiều đã cho.

-1	0	-1	-1	-1
0	-1	-1	0	0
0	0	-1	-1	0
0	0	0	0	-1
0	-1	0	-1	0
0	0	-1	0	-1
-1	0	0	0	-1

Hình a: ma trận chỉ chứa giá trị 0 & -1

1	0	2	2	2
0	2	2	0	0
0	0	2	2	0
0	0	0	0	3
0	4	0	5	0
0	0	6	0	7
8	0	0	0	7

Hình b: đánh số thứ tự 8 miền liên thông của mảng a

3.1.5. Sử dụng hàm map hoặc filter trên list

34/. Cho list lst chứa số nguyên. Sử dụng hàm map để các số nguyên trong lst thành dạng chuỗi

Ví dụ: Original list: [1, 2, 3, 4]

List of strings: ['1', '2', '3', '4']

35/. Cho 2 list B và E có số lượng phần tử bằng nhau. Sử dụng hàm map để tạo ra 1 list mới với kết quả là B^E của từng cặp phần tử

Ví dụ: bases_num = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

exponential = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

result= [10, 400, 27000, 2560000, 312500000, 46656000000, 8235430000000, 1677721600000000, 387420489000000000, 1000000000000000000]

36/. Cho 2 list *lstX* và *lstY* chứa các số nguyên và có cùng số lượng phần tử.

- Viết hàm *Addition_Subtraction* với tham số truyền cho hàm là 2 số nguyên x và y . Hàm trả về 2 giá trị nguyên là $x-y$ và $x+y$.
- Trong chương trình chính, sử dụng hàm map để tạo ra 1 list mới chứa các bộ 2 số do hàm *Addition_Subtraction* trả về khi nhận các giá trị của 2 list *lstX* và *lstY*.

Ví dụ: với 2 list: `listX = [6, 5, 3, 9]` và `listY = [0, 1, 7, 7]`

Sẽ xuất ra: Kết quả thực hiện: `[(6, 6), (6, 4), (10, -4), (16, 2)]`

37/. Cho 1 list (*lst*) chứa các số nguyên.

- Viết hàm *FindNumber* với tham số truyền cho hàm là số nguyên k . Hàm trả về 3 giá trị nguyên là *pre*, k , *next*:
 - *pre* là số chính phương nhỏ hơn và gần với k nhất. Quy ước, nếu $k \leq 1$ thì *pre*=0.
 - *next* là số chính phương lớn hơn và gần với k nhất. Quy ước, nếu $k < 1$ thì *next*=1.
- Trong chương trình chính, sử dụng hàm map để tạo ra 1 list chứa các bộ 3 số do hàm *FindNumber* trả về khi nhận các giá trị của list *lst*.

Ví dụ: List ban đầu `lst = [8, -2, 5, 9]`

Sẽ xuất ra: Kết quả thực hiện: `[(4, 8, 9), (0, -2, 1), (4, 5, 9), (4, 9, 16)]`

38/. Cho người dùng nhập 1 danh sách tên các màu cách nhau bởi dấu phẩy (,), ví dụ: Red, Blue, Black, White, Pink. Viết chương trình cho biết những ký tự:

- Có xuất hiện trong danh sách (không phân biệt ký tự hoa/thường).
- Không xuất hiện trong danh sách (không phân biệt ký tự hoa/thường).

✎ Ví dụ:

Danh sách ban đầu: `['Red', 'Blue', 'Black', 'White', 'Pink']`

Những ký tự xuất hiện trong list ban đầu: `a b c d e h i k l n p r t u w`

Những ký tự không có trong list ban đầu: `f g j m o q s v x y z`

✎ Gợi ý: sử dụng hàm map để chuyển mỗi chuỗi trong danh sách thành 1 list, từ đó mới xác định những ký tự có hoặc không có xuất hiện.

3.1.6. Sử dụng một số module trong việc xử lý trên list

3.1.6.1. Itertools module

39/. Viết chương trình cho nhập 1 chuỗi (S) và số nguyên (n). In ra tất cả các hoán vị (permutations) n phần tử từ các ký tự có trong S.

Ví dụ: với `S = 'xyz'` và `n=2`

Sẽ in ra: `[('x', 'x'), ('x', 'y'), ('x', 'z'), ('y', 'x'), ('y', 'y'), ('y', 'z'), ('z', 'x'), ('z', 'y'), ('z', 'z')]`

✎ Gợi ý: sử dụng `product` trong module `itertools`

40/. Viết chương trình cho nhập 1 chuỗi (S). Thực hiện xóa tất cả các ký tự liên tiếp trùng nhau trong S.

Ví dụ: với `S = 'xxxxyxxByyAAyyy'` và `n=2`

Sẽ in ra: `S = xyxByAy`

✎ Gợi ý: sử dụng `groupby` trong module `itertools`

3.1.6.2. Collections module

41/. Viết chương trình cho nhập 1 chuỗi (S). Thực hiện tách thành 2 chuỗi S1 và S2, với S1 chỉ chứa các ký tự xuất hiện 1 lần trong S, S2 chứa các ký tự có số lần xuất hiện trong S từ 2 lần trở lên.

Ví dụ: với `S= 'mommy and daddy'`

Sẽ in ra: `S1= no`

`S2= uadmy`

42/. Cho 3 list chứa các số nguyên L1, L2, L3. Các số trong mỗi list trên có thể trùng nhau và không được sắp xếp thứ tự. Viết chương trình để so sánh 2 list L1 với L2 và L3 với L2.

Lưu ý: không chuyển đổi list sang kiểu set để thực hiện vì set chỉ loại bỏ các giá trị trùng nhau nhưng không thể đếm được số lượng của từng giá trị trong list.

Ví dụ: cho 3 list sau:

Tên	Thành phần	Đếm và phân loại các giá trị	Kết quả so sánh
L1	[20, 10, 30, 10, 20, 30]	{10: 2, 20: 2, 30: 2}	Khác nhau
L2	[30, 20, 10, 30, 20, 50]	{10:1, 20:2, 30:2, 50:1}	Giống nhau
L3	[10, 20, 20, 30, 30, 50]	{10:1, 20:2, 30:2, 50:1}	Giống nhau

3.2. Tuple

43/. Cho nhập 1 dãy số cách nhau bởi dấu phẩy. Chương trình phát sinh ra 1 tuple từ dãy số trên.

Ví dụ: nhập '1, 3, 9, 2, 8, 4, 7' \Rightarrow tuple=(1, 3, 9, 2, 8, 4, 7).

44/. Viết chương trình thực hiện lần lượt các yêu cầu sau:

- Tạo 2 tuple: *tupleA* chứa 3 số nguyên, *tupleB* chứa 4 số nguyên.
- Tạo *tupleC* bằng cách kết hợp từ *tupleA* và *tupleB*.
- Tạo tiếp *tupleD* từ *tupleC* với các phần tử được sắp xếp giảm dần.
- In ra 2 phần tử đầu và 2 phần tử cuối của *tupleD*.

45/. Lần lượt cho người dùng nhập một số chuỗi bất kỳ. Các chuỗi được nhập sẽ được đưa vào 1 tuple. Việc nhập kết thúc khi người dùng nhập chuỗi rỗng (chỉ nhấn phím Enter mà không nhập). Yêu cầu:

- Gọi *n* là số lượng chuỗi vừa nhập. Cho nhập tiếp 1 số nguyên *k* ($-n \leq k < n$). In ra giá trị của phần tử có *index=k*.

Ví dụ: `animalTuple=('ant', 'bear', 'cat', 'dog', 'elephant', 'fish', 'goat', 'hippo')`, với *k*=-3 hoặc *k*= 5 sẽ xuất ra *fish*.

- Nhập chuỗi cần tìm (*sFind*).

(i)- Tìm và đếm số lần xuất hiện của *sFind* trong tuple.

(ii)- Nếu trong tuple có chứa nhiều *sFind* (trùng nhau), hãy in ra tất cả các vị trí tìm thấy chuỗi *sFind*

46/. Viết chương trình thực hiện lần lượt các yêu cầu sau:

- Lần lượt cho người dùng nhập một số lượng số nguyên bất kỳ. Các số được nhập sẽ được đưa vào 1 tuple (*aTuple*). Việc nhập kết thúc khi người dùng nhập giá trị zero (0).
- Thực hiện tương tự để tạo thêm 1 tuple khác (*bTuple*).
- Tạo mới 1 tuple (*cTuple*) bằng cách kết hợp 2 *bTuple* và *aTuple* sao cho *cTuple* không chứa giá trị trùng nhau. In ra màn hình các thành phần có trong *cTuple*.
- Tạo mới 1 tuple (*dTuple*) từ *cTuple* với các phần tử được sắp xếp giảm dần. In ra màn hình các thành phần có trong *dTuple*.
- In các phần tử trong *dTuple* có chỉ số (*index*) là số lẻ.

3.3. Dictionary

3.3.1. Xử lý trên dictionary

47/. Cho người dùng nhập 1 chuỗi (S).

- Đếm số lượng ký số, số lượng ký tự có trong S.
- Tính tổng các ký số có trong S.

Ví dụ: `S = 'Python 3.9.1' ⇒ 13`

48/. Cho người dùng nhập 1 chuỗi (S).

- Tìm tất cả ký tự trong chuỗi chỉ xuất hiện 1 lần.
- Tìm ký tự đầu tiên trong chuỗi chỉ xuất hiện 1 lần.

Ví dụ: `S = 'abcdef' ⇒ a`

`S = 'abcabcdef' ⇒ d`

`S = 'aabbcc' ⇒ None`

d

- Cho nhập 1 chuỗi (S). Tìm ký tự đầu tiên được lặp lại trong S.

Ví dụ: `S = "abcabc" sẽ in ra a`

`S = "abbcabc" sẽ in ra b`

`S = "abc" sẽ in ra None`

49/. Lần lượt thực hiện bài tập này bằng 2 cách sử dụng tuple và dictionary. Cho nhập một chuỗi (s), cho biết các kết quả sau (lưu ý 1 từ không được chứa các ký tự xuống dòng ('\n') hoặc các dấu phẩy (,), dấu chấm (.), vì vậy cần thực hiện loại bỏ chúng ra khỏi các từ trước khi thực hiện các yêu cầu):

- Tần suất xuất hiện của mỗi từ
- Các từ có số lần xuất hiện nhiều nhất
- Từ dài nhất.

Ví dụ: với `s = "Chiều chiều trước bến Văn Lâu.`

`Ai ngồi, ai câu, ai sầu, ai thảm.`

`Ai thương, ai cảm, ai nhớ, ai trông.`

`Thuyền ai thấp thoáng ven sông.`

`Đưa câu mái vẩy chạnh lòng nước non."`

Cho kết quả:

YÊU CẦU a: Số lần xuất hiện của các từ (Đã loại những từ trùng nhau):

```
{('non', 1), ('sầu', 1), ('trước', 1), ('thấp', 1), ('Văn', 1), ('vẩy', 1), ('chiều', 1), ('ven', 1), ('chạnh', 1), ('cảm', 1), ('nhớ', 1), ('Đưa', 1), ('thảm', 1), ('thương', 1), ('lòng', 1), ('ngồi', 1), ('Thuyền', 1), ('trông', 1), ('mái', 1), ('thoáng', 1), ('ai', 7), ('bến', 1), ('Ai', 2), ('nước', 1), ('Lâu', 1), ('Chiều', 1), ('sông', 1), ('câu', 2)}
```

YÊU CẦU b: Các từ có số lần xuất hiện nhiều nhất (7 lần) là: ai

YÊU CẦU c: Từ dài nhất là thoáng, Thuyền, thương, gồm 6 ký tự

➤ Gợi ý cách thực hiện bằng tuple:

B1. Dựa vào khoảng trắng giữa các từ, tách các từ đưa vào `list1`

B2. Trong các từ vừa tách vẫn còn lẫn ký tự xuống dòng ('\n') hoặc các dấu phẩy, dấu chấm trong ngắt câu. Hãy tìm cách loại bỏ chúng.

B3. Tạo ra 1 list đếm số lần xuất hiện của các từ đưa vào `list2`

B4. Sử dụng hàm zip với tham số là `list1` và `list2` để tạo thành các tuple. Đưa tất cả các tuple này vào `list3`.

B5. Do `list3` vẫn còn chứa các tuple có giá trị trùng nhau, nên dùng tiếp hàm set để loại bỏ các tuple có nội dung trùng nhau. Đây là kết quả cần thực hiện

Gợi ý sử dụng dictionary:

- B1.** Dựa vào khoảng trắng giữa các từ, tách các từ đưa vào *list* ()
- B2.** Trong các từ vừa tách vẫn còn lẫn ký tự xuống dòng ('\n') hoặc các dấu phẩy, dấu chấm trong ngắt câu. Hãy tìm cách loại bỏ chúng
- B3.** Khai báo 1 dictionary để đếm số lần xuất hiện của các từ.
- B4.** Để tìm từ dài nhất, sử dụng hàm *len*(chuỗi của từng có trong list)

Số strobogrammatic

Số strobogrammatic

- Là một số có giá trị không đổi khi xoay số đó 180 độ (180°). Nhận xét: các số *strobogrammatic* chỉ chứa các số sau đây: 0, 1, 6, 8, 9. Trong đó các số 0, 1, 8 không bị thay đổi giá trị sau khi xoay, còn số 6 và 9 bị thay đổi (6 chuyển thành 9 và 9 chuyển thành 6).

Ví dụ 1: số 619 sau khi xoay vẫn sẽ là 619 => 619 là số *Strobogrammatic*

Ví dụ 2: số 68910 sau khi xoay sẽ là 01689 => 68910 không phải là số

Strobogrammatic vì 68910 ≠ 01689

- Các số *strobogrammatic* đầu tiên là 0, 1, 8, 11, 69, 88, 96, 101, 111, 181, 609, 619, 689, 808, 818, 888, 906, 916, 986, 1001, 1111, 1691, 1881, 1961, 6009, 6119, 6699, 6889, 6969, 8008, 8118, 8698, 8888, 8968, 9006, 9116, 9696, 9886, 9966, ...

Số nguyên tố strobogrammatic

- Là số vừa là số nguyên tố, vừa là số *strobogrammatic*.
- Các số nguyên tố *strobogrammatic* đầu tiên là: 11, 101, 181, 619, 16091, 18181, 19861, 61819, 116911, 119611, 160091, 169691, 191161, 196961, 686989, 688889, ...

Số strobogrammatic mở rộng

Khi xoay các số dạng digital, ta thấy các số sau đây vẫn có ý nghĩa là 0, 1, 2, 5, 6, 8, 9. Trong đó các số 0, 1, 2, 5, 8 không bị thay đổi giá trị sau khi xoay, còn số 6 và 9 bị thay đổi (6 chuyển thành 9 và 9 chuyển thành 6)

Viết chương trình sử dụng đối tượng dictionary để thực hiện các yêu cầu sau:

- 50/. In ra các số strobogrammatic nhỏ hơn 1 triệu (1000000).
- 51/. In ra các số nguyên tố strobogrammatic nhỏ hơn 1 triệu (1000000).
- 52/. In ra các số strobogrammatic mở rộng nhỏ hơn 1 triệu (1000000).
- 53/. In ra các số nguyên tố strobogrammatic mở rộng nhỏ hơn 1 triệu (1000000).
- 54/. In ra các số nhỏ hơn 1 triệu (1000000) không phải là số strobogrammatic và không phải là số nguyên tố nhưng số strobogrammatic của số này lại là số nguyên tố.
- 55/. Cho dictionary *products* với nội dung như sau, trong đó mô tả tên sản phẩm và giá bán tương ứng. Yêu cầu:
 - a. Khai báo và gán giá trị cho dictionary trên
 - b. Viết hàm *Xuat(dictionary_Name)* để in nội dung dictionary_Name ra màn hình theo dạng: *key:value*. Ví dụ:

products

'SMART WATCH'	: 550,
'PHONE'	: 1000,
'PLAYSTATION'	: 500,
'LAPTOP'	: 1550,
'MUSIC PLAYER'	: 600,
'TABLET'	: 400

Products list

```
'SMART WATCH' : 550,
'PHONE' : 1000,
'PLAYSTATION' : 500,
'LAPTOP' : 1550,
'MUSIC PLAYER' : 600,
'TABLET' : 400
```

c. Viết chương trình chính lần lượt gọi các hàm trên thực hiện.

56/. Cho 2 dictionary với nội dung như sau, trong đó PhongBan_dict mô tả mã số và tên phòng ban; NhanVien_dict mô tả mã số, họ tên, ngày vào làm việc, mã phòng ban mà người đó đang làm việc. Yêu cầu:

PhongBan_dict

```
101 : Nhân sự
102 : Tài Vụ
103 : Marketing
104 : Kinh doanh
105 : Kỹ thuật
106 : Chăm sóc khách hàng
```

NhanVien_dict

```
1000 : {'HoTen': 'Tý', 'NgàyVaoLam': '01-10-89', 'MaPB': 103}
1001 : {'HoTen': 'Sửu', 'NgàyVaoLam': '01-11-88', 'MaPB': 101}
1002 : {'HoTen': 'Dần', 'NgàyVaoLam': '01-10-87', 'MaPB': 104}
1003 : {'HoTen': 'Mẹo', 'NgàyVaoLam': '01-06-89', 'MaPB': 105}
1004 : {'HoTen': 'Thìn', 'NgàyVaoLam': '01-01-86', 'MaPB': 106}
1005 : {'HoTen': 'Ty', 'NgàyVaoLam': '01-02-89', 'MaPB': 101}
```

a. Khai báo và gán giá trị cho 2 dictionary trên

b. Viết hàm Xuat(dictionary_Name) để in nội dung dictionary_Name ra màn hình theo dạng: key : value. Ví dụ:

```
101 : Nhân sự
102 : Tài Vụ
103 : Marketing
104 : Kinh doanh
105 : Kỹ thuật
106 : Chăm sóc khách hàng
```

c. Viết hàm TimNhanVien (MaNV) để tìm nhân viên có mã số là MaNV. Nếu tìm thấy trả về dictionary chứa thông tin nhân viên (gồm HoTen, NgàyVaoLam, MaPB), ngược lại khi không tìm thấy, trả về None.

d. Viết chương trình chính lần lượt gọi các hàm trên thực hiện để xuất lần lượt các yêu cầu sau:

(i). Gọi hàm Xuat để in danh sách phòng ban và danh sách nhân viên ra màn hình.

(ii). Cho nhập mã số nhân viên cần tìm (mMaNV), gọi hàm TimNhanVien để tìm. Nếu tìm thấy, sử dụng hàm Xuat để in thông tin của nhân viên có mã số là mMaNV ra màn hình, ngược lại in thông báo 'Không tìm thấy mMaNV'.

(iii).(*) Thực hiện tương tự câu (ii), nhưng viết thêm hàm Xuat2, hàm sẽ in thêm mã nhân viên và tên phòng ban của nhân viên đó. Ví dụ sau khi tìm nhân viên có mã 1001, sẽ in ra màn hình như sau:

```
Ma NV      : 1001
HoTen      : Sửu
NgàyVaoLam : 01-11-88
MaPB       : 101
Ten Phong Ban : Nhân sự
```

57/. Tổ chức chương trình quản lý danh bạ điện thoại (giả sử tên người là không trùng nhau và 1 người chỉ có 1 số điện thoại) bằng đối tượng dictionary, với các cặp **key-value** là **Tên-Số điện thoại**. Tạo hệ thống menu cho chương trình như sau:

CHƯƠNG TRÌNH QUẢN LÝ DANH BẠ ĐIỆN THOẠI

- 1.- Thêm mới 1 tên cùng số điện thoại
- 2.- Cập nhật lại số điện thoại thông qua tên
- 3.- Tìm kiếm số điện thoại thông qua tên
- 4.- In toàn bộ danh bạ
- 5.- Xóa 1 tên (cùng số điện thoại)
- 0.- Kết thúc

Bạn chọn chức năng: __

Mình họa danh bạ điện thoại

Tên	Số điện thoại
Tý	0123456789
Sửu	1234567890
Dần	9876543210

58/. Viết lại chương trình trò chơi “Bao-Búa-Đỉnh” cho 2 người chơi bằng cách sử dụng Dictionary. Mỗi người dùng được chọn 1 trong 3 món Bao, búa, đỉnh. Chương trình cho biết người thắng cuộc theo quy ước:

- Bao thắng Búa
- Búa thắng Đỉnh
- Đỉnh thắng Bao

59/. Viết chương trình cho người dùng nhập ngày (d) và tháng (m). In ra thứ của ngày tháng vừa nhập ở năm hiện tại.

☞ Gợi ý:

- Khai báo 1 dictionary với nội dung như sau:

```
weekDict = {1:'Thứ Hai',2:'Thứ Ba',3:'Thứ Tư',4:'Thứ Năm',
            5:'Thứ Sáu', 6:'Thứ Bảy',7:'Chủ nhật'}
```

- Sử dụng hàm `isoweekday` của module `datetime` và dựa vào `dictionary` này để in ra thứ.

60/. (*) Cho trước 1 dictionary như sau:

```
{ '1': 'abc', '2': 'def', '3': 'ghi', '4': 'jkl', '5': 'mno',
  '6': 'pqrs', '7': 'tuv', '8': 'wxy', '9': 'z' }
```

Viết chương trình cho nhập 1 chuỗi `s` chứa ký số (như '3', '15', '914', ...). In ra tất cả các tổ hợp có được từ việc lấy trong mỗi tổ hợp 1 phần tử. Ví dụ:

Chuỗi nhập	Kết quả
'1'	['a', 'b', 'c']
'29'	['dz', 'ez', 'fz']
'112'	['aad', 'aae', 'aaf', 'abd', 'abe', 'abf', 'acd', 'ace', 'acf', 'bad', 'bae', 'baf', 'bbd', 'bbe', 'bbf', 'bcd', 'bce', 'bcf', 'cad', 'cae', 'caf', 'cbd', 'cbe', 'cbf', 'ccd', 'cce', 'ccf']

☞ Gợi ý: giả sử chuỗi nhập là `s='27'`

- Đầu tiên tạo `resultList` chứa 1 chuỗi rỗng.
- Sử dụng `tempList` cho các lần phát sinh các tổ hợp sẽ được thực hiện sau đây. Khởi đầu mỗi lần lặp, `tempList` được gán là rỗng. Sau mỗi lần lặp, sẽ gán nội dung có trong `tempList` cho `resultList`.

- Lần lượt duyệt từng chữ số X trong chuỗi (S). Trong mỗi chuỗi số X thường gồm 3 ký tự, mỗi ký tự này sẽ kết hợp với các chuỗi đang có trong resultList để tạo thành các bộ chuỗi mới.

Mình họa: S='27'

- o resultList = ['']
- o tempList=[]
- o Xét chữ số đầu tiên (số 2). Chữ số 2 gồm 3 ký tự 'd', 'e', 'f'.
- o Lấy mỗi ký tự trong chữ số 2 để nối với chuỗi đang có trong resultList để có các phần tử mới trong \Rightarrow tempList = [''+'d', ''+'e', ''+'f'] = ['d', 'e', 'f']
- o Gán resultList = tempList
- o Gán lại tempList=[]
- o Xét số kế tiếp (số 7). Chữ số 7 gồm 3 ký tự 't', 'u', 'v'.
- o Tạo 1 hoặc nhiều newList bằng cách kết hợp các ký tự có trong số 7 với các subList đang có trong resultList \Rightarrow tempList=['t'+ 'd', 't'+ 'e', 't'+ 'f', 'u'+ 'd', 'u'+ 'e', 'u'+ 'f', 'v'+ 'd', 'v'+ 'e', 'v'+ 'f'] = ['dt', 'du', 'dv', 'et', 'eu', 'ev', 'ft', 'fu', 'fv']

61/. Cho người dùng nhập 1 chuỗi (S). Tìm từ trong S xuất hiện nhiều thứ hai (sau nhiều nhất).

🔗 Gợi ý: dùng hàm sorted trên dict, sau đó duyệt ngược dict để tìm số lớn thứ nhì.

Ví dụ: S='''Quê hương

Quê hương là chùm khế ngọt
Cho con trèo hái mỗi ngày
Quê hương là đường đi học
Con về rợp bướm vàng bay
Đố Trung Quân'''

Sẽ in ra ('là', 2) # vì 2 từ 'Quê' = 'hương' = 3

62/. Cho người dùng nhập 1 chuỗi (S). Thực hiện xóa tất cả những ký tự đi sau và bị trùng (đã xuất hiện trước đó (không phân biệt ký tự hoa/thường)).

Ví dụ: S='madam' \Rightarrow 'mad'

S='Collections' \Rightarrow 'coletins'

Gợi ý: sử dụng phương thức join của kiểu dữ liệu chuỗi với tham số là 1 dictionary được tạo từ chuỗi S.

63/. Viết chương trình sử dụng dictionary để quản lý danh bạ điện thoại với các cặp key-value được minh họa như sau:

Name	Telephone number
Johnny	0989741258
Katherine	0903852147
Misu	0913753951
Jack	0933753654
...	...

Yêu cầu:

- Tạo menu có dạng như hình minh họa sau. Sau khi người dùng chọn chức năng, chương trình thực hiện vừa chọn và in lại menu để người dùng tiếp tục sử dụng:

DANH BẠ ĐIỆN THOẠI
1.- Xem danh bạ
2.- Tìm kiếm theo tên
3.- Tìm kiếm theo số điện thoại
4.- Thêm mới
0.- Kết thúc

Bạn chọn chức năng cần thực hiện (0-4):

- b. Xem danh bạ: khi được chọn sẽ liệt kê toàn bộ tên và số điện thoại có trong danh bạ.
- c. Tìm kiếm theo tên: yêu cầu nhập tên, nếu tìm thấy, liệt kê tên và số điện thoại tương ứng. Nếu không tìm thấy, chương trình thông báo 'Không tìm thấy <tên>'.
- d. Tìm kiếm theo số điện thoại: yêu cầu nhập số điện thoại, nếu tìm thấy, liệt kê tên và số điện thoại tương ứng. Nếu không tìm thấy, chương trình thông báo 'Không tìm thấy <số điện thoại>'.
- e. Thêm mới: cho người dùng nhập tên và số điện thoại để bổ sung vào Danh bạ.
- f. Kết thúc chương trình.

64/. Tương tự như trên, viết chương trình sử dụng dictionary để quản lý từ điển Anh Việt với các cặp key là các từ tiếng Anh, value là nghĩa tiếng Việt tương ứng.

3.3.2. Sử dụng một số module trong việc xử lý trên dictionary

65/. Cho 1 tuple mà mỗi thành phần con trong đó cũng có kiểu là tuple, cho biết tên lớp cùng tên sinh viên. Ví dụ: `students = (('class_A', 'Tý'),`

```

('class_B', 'Suu'),
('class_A', 'Dan'),
('class_B', 'Mèo'),
('class_B', 'Thìn'),
('class_C', 'Tý'),

```

)

Yêu cầu: Đếm số lượng sinh viên của mỗi lớp.

66/. Cho trước 1 dictionary (ví dụ `mydict = {'Afghanistan' : 93, 'USA' : 213, 'Algeria' : 213, 'Angola' : 244, 'India' : 355, 'Albania' : 355, 'Andorra' : 376}`). Viết chương trình Python để tạo một thể hiện của `OrderedDict` dựa trên dictionary đã có. Sắp xếp từ điển trong quá trình tạo và in các thành phần của dictionary theo thứ tự ngược lại.

67/. Tương tự như trên, tổ chức chương trình từ điển (giả sử 1 từ tiếng Anh chỉ có 1 từ tiếng Việt tương ứng) bằng đối tượng dictionary, với các cặp **key-value** là **Tên-Số điện thoại**. Tạo hệ thống menu cho chương trình như sau:

CHƯƠNG TRÌNH TỪ ĐIỂN	
1.-	Thêm mới 1 cặp từ
2.-	Cập nhật lại nghĩa tiếng Việt
3.-	Tìm kiếm từ tiếng Anh
4.-	In toàn bộ từ điển
5.-	Xóa 1 từ
0.-	Kết thúc
Bạn chọn chức năng: __	

Mình họa từ điển

Từ Anh	Nghĩa Việt
Cat	Con mèo
Dog	Con chó
Bear	Con gấu

- a. In ra những ký tự xuất hiện trong cả 2 chuỗi.

Gợi ý: sử dụng class `Counter` trong module `collections` để chuyển mỗi chuỗi vào 1 dict thuộc class `Counter`. Thực hiện phép và (&) trên 2 dict này để có kết quả.

- b. Đếm xem có bao nhiêu ký tự có trong S1 nhưng không có trong S2 và có trong S2 nhưng không có trong S1.
- c. In ra những ký tự có trong S1 nhưng không có trong S2 và những ký tự có trong S2 nhưng không có trong S1.
- ☞ Gợi ý: đưa mỗi chuỗi vào 1 dict (dict1 và dict2). Thực hiện dò tìm S1 trên dict2 và tìm S2 trên dict1.

3.4. Set

68/. Tổ chức chương trình dưới dạng các hàm chức năng để thực hiện việc xử lý trên set như sau:

Yêu cầu:

- Viết hàm cho phép người dùng lần lượt nhập các phần tử số cho set cho đến khi giá trị nhận vào là -1. Hàm trả về set vừa nhập. Sử dụng hàm này để tạo ra 2 set: set1 và set2.
- In các số có trong set1 và set2 ra màn hình.
- Cho biết mỗi set có bao nhiêu phần tử, tổng giá trị các phần tử của mỗi set.
- Tìm giá trị lớn nhất, nhỏ nhất của mỗi set.
- Tìm những số có trong set1 hoặc set2.
- Tìm những số có trong cả 2 set set1 và set2.
- Tìm những số có trong set1 nhưng không có trong set2.
- Tìm những số có trong set1 hoặc set2 nhưng số đó không được có trong cả 2 set.
- Sắp xếp set1 tăng dần và set2 giảm dần.

69/. Viết chương trình tạo hệ thống menu với những chức năng như sau:

BÀI TẬP THỰC HÀNH VỀ SET

- 1.- Nhập giá trị cho set1
 - 2.- Nhập giá trị cho set2
 - 3.- Tính tổng các phần tử có trong mỗi set
 - 4.- Tìm giá trị lớn nhất trong mỗi set
 - 5.- Tìm giá trị nhỏ nhất trong mỗi set
 - 6.- Xóa phần tử có trong set1
 - 7.- Tìm và in ra những phần tử trùng nhau của 2 set
 - 8.- Tìm và in ra những phần tử KHÔNG trùng nhau của 2 set
 - 9.- In ra những phần tử có trong set1 nhưng không có trong set2
 - 10.- In ra những phần tử có trong set2 nhưng không có trong set1
 - 11.- Sắp xếp set1 giảm dần và set2 tăng dần
 - 0.- Kết thúc chương trình
- Bạn chọn chức năng số: _

☞ Yêu cầu:

- Chức năng 1 và 2: khi người dùng nhập số đã có trong set, yêu cầu nhập lại số khác cho đến khi không còn bị trùng.
- Chức năng 1: việc nhập kết thúc khi giá trị nhập vào là zero (0).
- Chức năng 2: hỏi người dùng cần nhập mấy số, chương trình cho nhập lần lượt từng số.
- Chức năng 6: cho người dùng nhập số cần xóa. Tìm và xóa giá trị này, sau đó in ra set sau khi xóa.
- Chức năng 8: loại bỏ những số trùng nhau trong kết quả (nếu có)
- Chương trình chỉ kết thúc khi giá trị nhập là 0.
- Nếu giá trị không phải các giá trị từ 0-11, chương trình hiện ra câu nhắc người dùng phải nhập đúng.

3.5. Phối hợp các đối tượng dạng sequence type

3.5.1. List & Set

70/. Viết chương trình tạo 1 list L gồm n phần tử có giá trị ngẫu nhiên từ 0 đến 10, với n do người dùng nhập. Tính tổng các số trong list sau khi đã loại bỏ các số trùng nhau (chỉ giữ lại 1 số trong các số trùng nhau).

Ví dụ: `L=[1, 2, 2, 2, 3, 5, 5] ⇒ tổng =11`

71/. Cho người dùng nhập 1 danh sách tên các màu cách nhau bởi dấu phẩy (,), ví dụ: Red, Blue, Black, White, Pink.

- Viết chương trình loại bỏ màu đầu tiên có trong list L
- Viết chương trình sắp xếp danh sách tên các màu theo thứ tự alphabet tăng dần. Nếu có những màu trùng nhau thì chỉ giữ lại 1 màu đại diện cho nhóm bị trùng.

```
color = input("Input comma separated sequence of words").split(',')
print(", ".join(sorted(list(set(color)))))
```

3.5.2. String & Set

72/. Cho người dùng nhập 1 chuỗi (S). Cho biết S có chứa đầy đủ tất cả các ký tự từ A-Z hay không? Không phân biệt ký tự hoa/thường.

☞ Gợi ý:

- Tạo ra 1 set chứa tất cả các ký tự thường từ a-z bằng 1 trong 2 cách sau:
 - Cách 1: tự khai báo 1 set trong đó liệt kê đầy đủ các ký tự từ a-z
 - Cách 2: sử dụng thuộc tính `ascii_lowercase` trong module `string`.
- Chuyển tất cả các ký tự trong S thành ký tự thường.
- Sử dụng 1 trong các toán tử so sánh (`>`, `>=`, `<`, `<=`, `==`, ...) để tập hợp vừa có, nếu bằng nhau sẽ cho ra `True`, ngược lại sẽ cho `False`.
- Sử dụng 2 chuỗi sau để kiểm tra:
 - `The quick brown fox jumps over the lazy dog` (đủ các ký tự từ a-z)
 - `The quick brown fox jumps over the lazy cat` (thiếu 2 ký tự d và g)

73/. Viết chương trình cho nhập số điện thoại (S). In ra các số từ 0 đến 9 không xuất hiện trong số điện thoại vừa nhập.

Ví dụ: nhập `S='0913158020'`

⇒ Trong số điện thoại 0913158020 không chứa các ký số: `[4, 6, 7]`

☞ Gợi ý:

- Tạo set1 chứa tất cả các ký số từ 0-9. Tương tự bài tập trước, cũng có 2 cách để tạo set:
 - Cách 1: tự khai báo 1 set trong đó liệt kê đầy đủ các ký số từ 0-9
 - Cách 2: sử dụng thuộc tính `string.digits` trong module `string`.
- Tạo set2 chứa tất cả các ký số có trong số điện thoại.
- Sử dụng phép bù giữa set1 và set2 để tìm kết quả

74/. Cho nhập 1 chuỗi (S). Tìm từ đầu tiên lặp lại trong S.

```
Ví dụ: S="ab ca bc ab"           sẽ in ra ab
       S="ab ca bc ca ab bc"     sẽ in ra ca
       S="ab ca bc"              sẽ in ra None
```

☞ Gợi ý: Tạo set1 rỗng. Lần lượt đưa từng ký tự của S vào trong set, nếu ký tự nào đã có trong set1 thì kết thúc. Sử dụng toán tử `in` để kiểm tra ký tự đã có trong set1 hay chưa?

75/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Tìm đoạn ngắn nhất trong S sao cho đoạn này chứa tất cả các ký tự có trong S.

Ví dụ: S1 = 'how can i tell her-lobo'=> in ra: 'w can i tell her-lob'

↳ Gợi ý: sử dụng defaultdict trong module collections

3.5.3. List & Dictionary

76/. Cho nhập 1 số nguyên n, cho biết số lượng từng ký số có trong n. Lưu ý: chỉ in ra các ký số có xuất hiện. Yêu cầu sử dụng list để lưu biến đếm.

Ví dụ: n=1008 => in ra Số 1008 gồm:

Số 0 có 2 số

Số 1 có 1 số

Số 8 có 1 số

77/. Viết chương trình nhập vào năm. In ra tên của năm âm lịch tương ứng.

Ví dụ: nhập năm 2020 in ra Canh Tý.

Yêu cầu: lần lượt thực hiện bài tập bằng cách sử dụng các cấu trúc list, tuple và dictionary

Biết rằng:

CAN	Giáp	At	Bính	Đinh	Mậu	Kỷ	Canh	Tân	Nhâm	Quý		
CHI	Tý	Sửu	Dần	Mão	Thìn	Tỵ	Ngọ	Mùi	Thân	Dậu	Tuất	Hợi

78/. Lần lượt thực hiện các yêu cầu sau:

- Viết hàm cho người dùng nhập 1 số nguyên n với $1 < n \leq 100$. Do đó nếu n không phải là số nguyên và nếu n không nằm trong khoảng từ 2->100, chương trình sẽ yêu cầu nhập lại cho đến khi đúng yêu cầu.
- Viết hàm tạo 1 list L gồm n phần tử với giá trị được phát sinh ngẫu nhiên trong khoảng từ Min=-5 đến Max=+10. Lưu ý các giá trị này có thể là số nguyên hoặc số thực.
- Viết hàm đếm tần suất xuất hiện của các giá trị có trong list L. Sau đó cho biết số tần suất lớn nhất là bao nhiêu và những giá trị nào có tần suất là nhiều nhất.
- Viết hàm kiểm tra kiểu dữ liệu của từng phần tử trong list L. In ra số lượng phần tử có kiểu dữ liệu là số nguyên, bao nhiêu phần tử có kiểu dữ liệu là số thực.
- Sắp xếp list L tăng dần.
- Viết hàm tính trung bình các phần tử trong list L.

3.5.4. List & Tuple

79/. Cho một list mà mỗi thành phần trong đó có kiểu là 1 tuple. Ý nghĩa của mỗi tuple là thành phần thứ nhất cho biết tên của lớp học, thành phần thứ 2 cho biết tên của sinh viên.

Viết chương trình để nhóm các tuple thành 1 dictionary với key là tên lớp và value là 1 list chứa tên các sinh viên. In ra kết quả được sắp xếp theo tên lớp

Ví dụ: studentsList = [('classB', 'Sửu'), ('classA', 'Dần'), ('classC', 'Mẹo'), ('classB', 'Thìn'), ('classA', 'Tý'), ('classB', 'Tỵ')]

Kết quả thực hiện:

```
[('classA', ['Dần', 'Tý']), ('classB', ['Sửu', 'Thìn', 'Tỵ']), ('classC', ['Mẹo'])]
```

3.6. Sử dụng hàm map để chuyển đổi giữa các đối tượng dạng danh sách

3.6.1. List to List

80/. *Interleave two given list to another list randomly*: Cho 2 list chứa các số nguyên, với số lượng phần tử có trong 2 list là khác nhau. Yêu cầu tạo ra 1 list mới từ 2 list ban đầu sao cho các phần tử của 2 list ban đầu được xếp xen kẽ nhau một cách ngẫu nhiên.

Ví dụ: cho 2 list ban đầu:

```
lst1 = [1,2,3,4,5,6]
lst2 = [7,8,9,10,11,12,13]
```

Sau khi hoàn tất, ta sẽ thu được:

```
lst3 = [1, 7, 8, 9, 2, 10, 11, 3, 4, 12, 5, 13, 6]
Hoặc lst3 = [7, 1, 2, 3, 4, 8, 9, 5, 10, 11, 6, 12, 13]
```

➤ Gợi ý: sử dụng hàm iter để tạo đối tượng lặp (iterator object)

3.6.2. Dictionary of List to List of Dictionary:

81/. Cho 1 dictionary với kiểu dữ liệu của key là chuỗi và kiểu dữ liệu của value là 1 list chứa các số nguyên. Thực hiện chuyển mỗi thành phần trong dictionary ban đầu thành nhiều các dictionary mới, trong đó key của dictionary mới chính là key của dictionary ban đầu và value là các giá trị có trong list ban đầu. Tất cả các dictionary kết quả sẽ được chứa trong cùng 1 list

Ví dụ: cho list ban đầu:

```
marks = {'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}
```

Sau khi hoàn tất, ta sẽ thu được:

```
afterList = [{'Science': 88, 'Language': 77},
             {'Science': 89, 'Language': 78},
             {'Science': 62, 'Language': 84},
             {'Science': 95, 'Language': 80}]
```

3.6.3. String to List

82/. Cho 1 list mà mỗi thành phần của list là 1 chuỗi. Thực hiện tách các chuỗi thành phần trong list mỗi chuỗi sẽ chuyển thành 1 sublist và mỗi ký tự trong chuỗi trở thành 1 thành phần trong sublist vừa có.

Ví dụ: cho list ban đầu:

```
beforeList = ["Red", "Green", "Black", "Orange"]
```

Sau khi hoàn tất, ta sẽ thu được:

```
afterList = [['R','e','d'], ['G','r','e','e','n'],
             ['B','l','a','c','k'], ['O','r','a','n','g','e']]
```

3.6.4. Tuple to List

83/. Cho 1 list mà mỗi thành phần của list là 1 tuple với thành phần trong tuple là các kiểu dữ liệu chuỗi.

- a. Thực hiện chuyển đổi để có 1 list mới mà mỗi thành phần trong đó là 1 chuỗi, mỗi chuỗi này được nhập lại từ các thành phần trong tuple ban đầu.

Ví dụ: cho list ban đầu:

```
beforeList = [('red', 'pink'), ('white', 'black', 'yellow'), ('orange', 'green')]
```

Sau khi hoàn tất, ta sẽ thu được

```
afterList = ['red pink', 'white black yellow', 'orange green']
```

- b. Thực hiện tương tự câu a, nhưng tách mỗi thành phần trong tuple thành 1 thành phần trong list.

Ví dụ: cho list ban đầu:

```
beforeList = [('red', 'pink'), ('white', 'black', 'yellow'), ('orange', 'green')]
```

Sau khi hoàn tất, ta sẽ thu được

```
afterList = ['red', 'pink', 'white', 'black', 'yellow', 'orange', 'green']
```

3.6.5. Others to String

84/. Cho 1 list chứa các số nguyên. Sử dụng lệnh map để chuyển list trên thành 1 list chứa chuỗi các số nguyên

Ví dụ: cho list ban đầu: `beforeList = [1, 2, 3, 4]`
 Sau khi hoàn tất, ta sẽ thu được `afterList = ['1', '2', '3', '4']`

85/. Thực hiện tương tự bài tập trên, nhưng thay list bằng 1 tuple

Ví dụ: cho tuple ban đầu: `beforeTuple = (1, 2, 3, 4)`
 Sau khi hoàn tất, ta sẽ thu được `afterTuple = ('1', '2', '3', '4')`

3.7. Xây dựng hàm ẩn danh (Anonymous Function) cho Iterator object

3.7.1. Xây dựng hàm ẩn danh trên number list

86/. Cho 2 list chứa các số nguyên và 2 list này có cùng số lượng phần tử. Sử dụng lambda để tạo ra 1 list mới bằng cách cộng đôi một các số có trong 2 list.

<code>lst1 =</code>	<code>[1, 2, 3]</code>
<code>lst2 =</code>	<code>[4, 5, 6]</code>
<code>resultList =</code>	<code>[5, 7, 9]</code>

⇒ Gợi ý: sử dụng hàm map

Ví dụ: `lst1 = [1, 2, 3]`, `lst2 = [4, 5, 6]` => in ra màn hình `[5, 7, 9]`

87/. Cho 3 list chứa các số nguyên có cùng số lượng phần tử. Tạo ra 1 list mới bằng cách chọn số lớn nhất trong bộ các phần tử có cùng thứ tự (index) có trong 3 list.

<code>lst1 =</code>	<code>[1, 2, 3, 9]</code>
<code>lst2 =</code>	<code>[4, 8, 6, 12]</code>
<code>lst3 =</code>	<code>[7, 5, 10, 11]</code>
<code>resultList =</code>	<code>[7, 8, 10, 12]</code>

Ví dụ: với 3 list cho trước: `lst1 = [1, 2, 3, 9]`, `lst2 = [4, 8, 6, 12]`, `lst3 = [7, 5, 10, 11]` => in ra màn hình `[7, 8, 10, 12]`.

88/. Cho list1 chứa các số nguyên bất kỳ. Sử dụng lambda để:

a. Từ list1, tạo list2 chứa các số chẵn, list3 chứa các số lẻ. In 2 list này ra màn hình.

Ví dụ `list1 = [-5, 10, -3, -1, 7, 8, 9, 2]`

In ra màn hình: List ban đầu: `[-5, 10, -3, -1, 7, 8, 9, 2]`

Các số chẵn có trong list: `[10, 8, 2]`

Các số lẻ có trong list: `[-5, -3, -1, 7, 9]`

b. Tương tự câu a nhưng lần lượt các lambda chỉ tính số lượng số chẵn, số lượng số lẻ. In kết quả thực hiện ra màn hình.

⇒ Gợi ý: sử dụng hàm filter khi tạo 2 list mới

c. Tạo list2 chứa các giá trị là bình phương của các giá trị có trong list1 và list3 chứa các giá trị là lũy thừa 3 của các giá trị có trong list1. In kết quả ra màn hình.

⇒ Gợi ý sử dụng hàm map khi tạo 2 list mới

Ví dụ

List ban đầu: `[-5, 10, -3, -1, 7, 8, 9, 2]`

Bình phương các giá trị trong list: `[25, 100, 9, 1, 49, 64, 81, 4]`

Lũy thừa 3 các giá trị trong list: `[-125, 1000, -27, -1, 343, 512, 729, 8]`

d. Tạo list2 chứa các giá trị là số nguyên tố hoặc số chính phương. Với số chính phương (square number hay perfect square) là số có căn bậc hai là một số nguyên. Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3.

⇒ Gợi ý viết 2 hàm kiểm tra số nguyên tố và kiểm tra số chính phương. Lambda sẽ sử dụng 2 hàm này trong điều kiện lọc của hàm filter

Ví dụ

List ban đầu: [19, 65, 81, 39, 152, 639, 121, 44, 100, 31]

Các số là số nguyên tố hoặc là số chính phương: [19, 81, 121, 100, 31]

89/. Viết chương *trình* cho nhập số lượng sinh viên (n). Cho nhập thông tin của n sinh viên, biết thông tin của mỗi sinh viên gồm tên và điểm. In ra tên và điểm của tất cả sinh viên có điểm thấp thứ nhì. Nếu có nhiều sinh viên cùng có điểm thấp thứ nhì, chương trình in tên theo thứ tự alphabet.

90/. Viết chương trình cho nhập số nguyên n. Chương trình tạo ra 1 list L gồm n số nguyên ngẫu nhiên từ 1 đến 10. Tính tích các phần tử có trong List bằng cách sử dụng lambda.

☞ Gợi ý: sử dụng hàm *reduce* trong module *functools*.

3.7.2. Xây dựng hàm ẩn danh trên string list

91/. Cho list1 chứa các chuỗi ký tự. Sử dụng lambda để tạo ra list2 mới thỏa mãn yêu cầu sau:

a. List2 chứa các chuỗi có chiều dài là 6.

☞ Gợi ý: sử dụng hàm *filter*.

Ví dụ: với List1 = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday']

Sẽ in ra: ['Monday', 'Friday', 'Sunday']

b. List2 chứa các chuỗi Palindrome. Chuỗi được gọi là Palindrome nếu sau khi đảo ngược các ký tự của nó, ta nhận được chuỗi ban đầu (chuỗi đối xứng). Ví dụ MADAM.

Ví dụ: với List1 = ['php', 'www', 'Python', 'abba', 'Java', 'MADAM']

Sẽ in ra: ['php', 'www', 'abba', 'MADAM']

3.7.3. Xây dựng hàm ẩn danh trên các đối tượng được phối hợp từ 2 loại iterator object khác nhau

92/. Cho cấu trúc chung của 1 tuple gồm 3 thành phần (đều là kiểu chuỗi) cho biết thông tin về sinh viên (họ tên, ngày sinh, trọng lượng tính theo kg). Cho trước 1 list mà mỗi thành phần của list là 1 tuple đã biết. Yêu cầu tách list ban đầu thành 3 list khác nhau, mỗi list chứa 1 thành phần của tuple: gồm list về họ tên, list về ngày sinh và list về trọng lượng. Tuy nhiên list về trọng lượng cần lược bỏ đơn vị “kg”.

Ví dụ: cho list ban đầu:

```
StudentList = [('Alberto Franco', '15/05/2002', '35kg'),
               ('Gino Mcneill', '17/05/2002', '37kg'),
               ('Ryan Parkes', '16/02/1999', '39kg'),
               ('Eesha Hinton', '25/09/1998', '35kg')]
```

Sau khi hoàn tất, ta sẽ thu được 3 list:

Student name: ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton']

Student dob: ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998']

Student weight: [35, 37, 39, 35]

93/. Cho 1 list mà các thành phần trong đó là 1 tuple. Ví dụ list ban đầu như sau:

```
subjectMarksList = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
```

Sử dụng lambda để sắp xếp list tăng theo điểm. Như vậy sau khi sắp xếp, **subjectMarksList** sẽ có dạng như sau:

```
[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]
```

94/. Cho 1 list mà các thành phần trong đó là 1 dictionary. Ví dụ list ban đầu như sau:

```
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]
```

Sử dụng lambda để sắp xếp list tăng theo key là 'color'. Như vậy sau khi sắp xếp, list sẽ có dạng như sau:

```
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}]
```

95/. Cho 1 list mà các thành phần trong đó là 1 tuple. Ví dụ list ban đầu như sau:

```
[('Phạm Tỷ', '15/05/2002', '35kg'), ('Đinh Sửu', '17/05/2002', '37kg'), ('Nguyễn Dân', '16/02/1999', '39kg'), ('Trần Mão', '25/09/1998', '35kg')]
```

Sử dụng hàm map kèm lambda để có kết quả như minh họa sau:

```
+ Danh sách sinh viên: [('Phạm Tỷ', '15/05/2002', '35kg'), ('Đinh Sửu', '17/05/2002', '37kg'), ('Nguyễn Dân', '16/02/1999', '39kg'), ('Trần Mão', '25/09/1998', '35kg')]
+ Tên các sinh viên: ['Phạm Tỷ', 'Đinh Sửu', 'Nguyễn Dân', 'Trần Mão']
+ Tuổi của các sinh viên: [18, 18, 21, 22]
+ Tuổi bình quân của các sinh viên: 19.75
+ Cân nặng bình quân: 36.5
```

3.7.4. Sử dụng module khác để xây dựng hàm ẩn danh trên iterator object

96/. Theo Wikipedia, đảo chữ (anagram) là hoán vị các ký tự có trong từ đó (các từ mới này có thể có nghĩa hoặc không có nghĩa). Ví dụ từ eat sẽ có thể được hoán vị thành các từ: tea, eta, tae, aet, ate.

Cho list1 chứa 1 số từ. Viết chương trình yêu cầu người dùng nhập 1 từ (w), sử dụng lambda để tìm các trường hợp đảo ký tự của w có trong list1.

Ví dụ: Các chuỗi có trong list1: ['bcda', 'abce', 'cbda', 'cbea', 'adcb']

Các đảo chữ của 'abcd' trong list1: ['bcda', 'cbda', 'adcb']

3.8. Sử dụng kỹ thuật Comprehension cho cho Iterator object

3.8.1. Number list comprehensions

97/. Viết chương trình cho nhập số nguyên n. Sử dụng comprehension để tạo ra 1 list chứa các ước số của n.

Ví dụ: với n=15, chương trình sẽ in ra: Các ước số của 15 là: [1, 3, 5, 15]

98/. Sử dụng comprehension để thực hiện các yêu cầu sau

a. Cho người dùng nhập lần lượt 8 số nguyên.

Yêu cầu: code sử dụng cho người dùng nhập các số, sử dụng lệnh *input* kèm với *for* theo dạng: `myList = [int(input("...")) for i in range(...)]`

b. In ra các số nguyên tố có trong 8 số vừa nhập.

99/. Giả sử có định nghĩa về số đồng nhất như sau: số k được gọi là số đồng nhất khi $k > 0$ và các ký số có trong k đều giống nhau, ví dụ: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 111, 222, 1111, ...

Yêu cầu: cho nhập số nguyên n ($n > 0$). Liệt kê các số đồng nhất nhỏ hơn n.

100/. Cho nhập nhiều số nguyên liên tiếp cách nhau bởi khoảng trắng. In ra các số này với thứ tự giảm dần.

☞ Yêu cầu: lần lượt thực hiện việc nhập các số theo 2 cách sau:

- Cách 1: `lst01 = list(map(int, input("...").split()))`
- Cách 2: `lst01 = [list(map(int, input("...").split()))]`

☞ Gợi ý khi sử dụng cách 2:

- Khi sử dụng Comprehension kết hợp với hàm map và phương thức split để nhập các giá trị cách nhau bởi ký tự khoảng trắng (hoặc ký tự khác). List tạo ra sẽ là 1 subList chứa trong 1 list khác. Vì vậy, cần làm phẳng list (chuyển sublist thành list), ta sử dụng theo cú pháp sau:

```
lst01 = [list(map(int, input("...").split()))]
#lst01 hiện chứa sublist => làm phẳng list bằng comprehension:
lst02=[x for sublist in lst1 for x in sublist]
```

- Dùng phương thức `sort()` với tham số `reverse` của `list`. Theo mặc định, tham số `reverse` trong phương thức `sort` sẽ có giá trị là `False` tức là `sort` tăng.

101/. Cho nhập 1 số nguyên n , tạo list L gồm n phần tử với giá trị ngẫu nhiên nguyên dương từ 0 đến 100000.

- a. Cho nhập số k . Sử dụng kỹ thuật Comprehension, cho biết k có lớn hơn tất cả các số có trong L hay không? Nếu không, chương trình xét tiếp xem k có lớn hơn bất kỳ số nào có trong list hay không? Nếu vẫn không có, chương trình in ra ' k nhỏ hơn tất cả các số có trong list'.

☞ Gợi ý: dùng hàm `all()` và `any()`.

- b. **Số giảm dần:** Giả sử n được gọi là số giảm dần khi giá trị các ký số có trong n giảm dần từ trái sang phải

Ví dụ: số giảm dần như: $n = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 21, 31, 32, \dots, 420, \dots, 77431, \dots$

số KHÔNG giảm dần như: $12, 13, \dots, 557, \dots, 77434, \dots$

☞ Yêu cầu cài đặt: Viết hàm liệt kê các số giảm dần có trong list.

☞ Kết quả gợi ý khi thực hiện chương trình:

Giả sử với list gồm `[727, 6, 1421, 626, 3706, 101, 3553, 4234, 33, 971]`

Hàm sẽ in ra kết quả: Trong list có 3 số giảm dần: `6, 33, 971`

Với list gồm `[1599, 1910, 14, 1050, 258, 4387]`

Hàm sẽ in ra kết quả: Trong list không chứa số giảm dần

102/. Tạo 2 listA, listB với kích thước khác nhau và chứa giá trị tùy ý. Ví dụ:

```
x = [1, 2, 3]
y = [5, 10, 15, 20]
```

Viết chương trình tạo ra listC bằng cách tính tích của từng phần tử trong x với từng phần tử trong y sao cho listC chỉ chứa những phần tử tích này nhưng loại đi những số chẵn. Ví dụ: khi lần lượt tính tích của từng phần tử trong x và từng phần tử trong y, ta được: `[5, 10, 15, 10, 20, 30, 15, 30, 45]`. Nhưng kết quả yêu cầu loại bỏ những phần tử chẵn nên listC chỉ còn: `[5, 15, 15, 45]`

103/. Tạo 1 list chứa các số nguyên tố nhỏ hơn 1000.

- ☞ Gợi ý:
- Tạo hàm nhận tham số là 1 số nguyên (k), hàm thực hiện kiểm tra xem k có là số nguyên tố hay không?
 - Sau đó trong lệnh comprehension sẽ gọi hàm này.

104/. Tạo 1 list chứa các số nguyên tố Palindrome nhỏ hơn 1000.

105/. Bài toán gà chó. Nội dung bài toán như sau:

Sử dụng kỹ thuật comprehension để viết chương trình tìm tất cả các nghiệm có thể có của bài toán. Trong mỗi trường hợp của kết quả, cho biết số lượng gà, số lượng chó?

Vừa gà vừa chó
Bó lại cho tròn
Đúng ba sáu con
Một trăm chân chẵn.

106/. Bài toán trăm trâu. Nội dung bài toán như sau:

Sử dụng kỹ thuật comprehension để viết chương trình tìm tất cả các nghiệm có thể có của bài toán. Trong mỗi trường hợp của kết quả, cho biết số lượng của mỗi loại trâu (đứng, nằm, già) có bao nhiêu con?

Trăm trâu trăm cỏ
Trâu đứng ăn năm
Trâu nằm ăn ba
Trâu già ăn một

107/. Cho nhập 1 dãy các hệ số nhị phân (0,1) ngăn cách nhau bởi dấu phẩy. In ra các số nhị phân trong dãy chia hết cho 5 (hệ thập phân). Kết quả in chính là dãy nhị phân đã nhập.

Ví dụ: s= '0000,00101,00110,010100,10101,11010,111100'
tương đương với các giá trị thập phân là 0, 5, 6, 20, 21, 31, 60
In ra 0000,00101,010100,111100
tương đương với các giá trị thập phân là 0, 5, 20, 60

108/. Tạo 3 listA, listB, listC với kích thước khác nhau và chứa giá trị tùy ý. Ví dụ: listA= [1, 2], listB= [1, 2, 3, 4], listC=[3, 4, 2]

Cho nhập tiếp số nguyên k. Viết chương trình tìm và in ra các tổ hợp gồm 3 phần tử (mỗi phần tử từ một list) có tổng = k.

🔗 **Gợi ý:**

- Sử dụng `itertools.product` để tạo các tổ hợp
- Sử dụng `partial` function cho phép sửa một số đối số nhất định của hàm và tạo một hàm mới.

Tổng dãy số

109/.

$$S = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+n}$$

110/. $S = 1 - \frac{1}{1+2} + \frac{1}{1+2+3} - \dots + (-1)^{n+1} \frac{1}{1+2+3+\dots+n}$

111/.

$$S = \frac{1}{1*2} + \frac{2}{2*3} + \frac{3}{3*4} + \dots + \frac{n}{n*(n+1)}$$

112/.

$$S = 1 + \frac{1+2}{2} + \frac{1+2+3}{3} + \dots + \frac{1+2+3+\dots+n}{n}$$

113/.

$$S = \frac{1^2 + 2^2 + 3^2 + \dots + n^2}{1*2 + 2*3 + 3*4 + \dots + n(n+1)}$$

114/. $S = x + x^2 + x^3 + \dots + x^n$

$$115/. S = x + x^3 + x^5 + \dots + x^{2n+1}$$

116/.

$$S = x + \frac{x^2}{1+2} + \frac{x^3}{1+2+3} + \dots + \frac{x^n}{1+2+3+\dots+n}$$

3.8.2. String list comprehensions

3.8.2.1. join method

117/. Cho nhập số nguyên dương n . In ra các số nguyên dương X nhỏ hơn hay bằng n , sao cho X có chứa ít nhất một số 5 ở bất kỳ vị trí nào (hàng chục, hàng đơn vị, hàng trăm, ...). Các số trong kết quả được in ra cách nhau bởi dấu phẩy (,).

Ví dụ: nhập $n=60$
In ra 5, 15, 25, 35, 45, 50

118/. Cho nhập số nguyên dương n . In ra các số nguyên dương (tạm gọi là X) nhỏ hơn hay bằng n , sao cho các ký số có trong X đều là số chẵn.

Ví dụ: nhập $n=29$
In ra 0, 2, 4, 6, 8, 20, 22, 24, 26, 28

119/. Viết chương trình cho nhập số 1 chuỗi (S). Chương trình thực hiện loại bỏ tất cả các nguyên âm (a, e, i, o, u) và là ký tự in thường có trong S.

Ví dụ: S = "Hello Everybody" \Rightarrow xuất ra Hll Evrybdy

120/. Cho nhập 1 chuỗi S. Chuyển tất cả các khoảng trắng nếu có ra phía đầu chuỗi S.

Ví dụ: (ký tự \cup được diễn tả thay cho khoảng trắng)
S = "uuw3resource.ucomu" \Rightarrow "uuuuuw3resource.com"

121/. Viết chương trình cho người dùng nhập 1 chuỗi S (có thể gồm nhiều dòng). Thực hiện đảo các từ trong chuỗi.

Ví dụ: S = 'Kiến ăn Cá' \Rightarrow xuất ra Cá ăn Kiến
Hay S = 'Chỉ có thuyền mới hiểu\nBiển mênh mông nhường nào'
 \Rightarrow xuất ra hiểu mới thuyền có Chỉ
nào nhường mông mênh Biển

122/. Giả sử người ta cho rằng 1 số gọi là số may mắn nếu chỉ chứa toàn các số 6 hoặc số 8. Viết hàm nhận tham số là số nguyên n , xét xem n có là số may mắn hay không?

Ví dụ: $n=686 \Rightarrow 686$ là số may mắn.
 $n=68626 \Rightarrow 68626$ KHÔNG phải số may mắn

123/. Cho người dùng nhập chuỗi (S) là 1 địa chỉ IP (IP Address). Thực hiện xóa các số 0 (zero) không có nghĩa.

Ví dụ: S = '192.024.001.023' \Rightarrow sẽ in ra 192.24.1.23
Hoặc S = '005.004.001.020' \Rightarrow sẽ in ra 5.4.1.20

3.8.2.2. String comprehension

124/. Viết chương trình cho người dùng nhập 1 chuỗi (S) bao gồm cả ký tự và ký số. Thực hiện tính tổng tất cả các ký số có trong S.

Ví dụ: S = 'Python 3.7' $\Rightarrow 10$

125/. Viết chương trình cho người dùng nhập 1 số nguyên n . Sử dụng kỹ thuật *comprehension* để xét xem các ký số trong n có tăng dần từ trái sang phải hay không?

Ví dụ: $n = 133577999 \Rightarrow$ TĂNG dần
 $n = 1335787999 \Rightarrow$ KHÔNG tăng dần

Gợi Ý: Sử dụng hàm *all*, bên trong hàm là câu lệnh có sử dụng kỹ thuật *comprehension*.
 Kết quả trả về của hàm *all* là *True* hoặc *False*.

126/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Đếm và in ra trong S có bao nhiêu ký tự là nguyên âm (vowel - **a, e, i, o, u**), bao nhiêu ký tự là phụ âm (consonant - **b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y, z**). Lưu ý khi đếm không phân biệt chữ hoa, chữ thường.

Ví dụ: S = 'Sai Gon' sẽ in ra Có 3 nguyên âm: a i o
 Có 3 phụ âm: S G n

127/. Viết chương trình cho người dùng nhập 2 chuỗi (S1 và S2). Tạo ra chuỗi kết quả bằng cách ghép các ký tự có trong S1 nhưng không có trong S2 và có trong S2 nhưng không có trong S1.

Ví dụ: S1='abcd'; S2='mncdgh' \Rightarrow 'abmng h'

3.8.3. Set comprehensions

128/. Cho 2 set: set1={1, 3, 5, 7, 9} và set2={0, 2, 4, 6, 8}. Tạo ra set3 chứa các số $x+y$ là số nguyên tố với x thuộc set1, y thuộc set2.

3.9. Giải quyết 1 yêu cầu bằng cả hai kỹ thuật Comprehension & Anonymous Function

Yêu cầu chung: Mỗi bài tập trong phần này sẽ được thực hiện lần lượt bằng cả 2 cách: Comprehension và Anonymous Function

129/. Viết chương trình cho nhập số nguyên n . Chương trình tạo ra 1 list L gồm n số nguyên âm hoặc dương ngẫu nhiên từ -100 đến +100. Tạo ra list M chỉ chứa những số trong L nhưng có giá trị >0 .

130/. Viết chương trình cho nhập 1 list gồm các số âm hoặc dương bất kỳ. Hãy đưa các số dương về đầu list, các số âm về cuối list. Lưu ý không sử dụng phương thức sort.

Ví dụ: với `lst = [-5, 10, -3, -1, 7, 8, 9, 2]`

Kết quả in ra màn hình: `[10, 7, 8, 9, 2, -5, -3, -1]`

✎ Gợi ý: tạo ra *listDuong* chỉ chứa số dương trong *lst* và tạo ra *listAm* chỉ chứa số âm trong *lst*. Sau đó nối 2 list này lại

131/. Viết chương trình cho nhập số nguyên n . Chương trình tạo ra 1 list L gồm n số nguyên ngẫu nhiên. In ra những số chia hết cho 15 từ list L.

132/. Tạo 2 listA, listB với kích thước khác nhau và chứa giá trị tùy ý. Ví dụ:

```
listA = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
listB = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

a. Viết chương trình trả về *listAinB* cho biết những giá trị nào trong *listA* có trong *listB* và ngược lại tạo 1 *listBinA* chứa những giá trị trong listB có trong listA.

b. (*) Mở rộng: Sao cho mỗi list kết quả (*listAinB* và *listBinA*) không chứa giá trị trùng nhau. Thực hiện bằng 2 cách: cách 1 dùng set; cách 2 dùng dictionary.

133/. Cho nhập số nguyên dương n . Kiểm tra xem các chữ số của n có toàn lẻ (hay toàn chẵn) không?


134/. Số may mắn: giả sử người ta cho rằng 1 số gọi là số may mắn nếu chỉ chứa toàn các số 6 hoặc số 8. Viết chương trình cho nhập số nguyên n , xét xem n có là số may mắn hay không?

Ví dụ: $n=686 \Rightarrow 686$ là số may mắn.

$n=68626 \Rightarrow 68626$ KHÔNG phải số may mắn.

XỬ LÝ NGOẠI LỆ (Exception Handling)

- 1/. Cho biến $x=2$ và biến y không gán giá trị. Viết chương trình kiểm tra xem biến x và y đã được khai báo và gán giá trị hay chưa?
- 2/. Cho người dùng nhập 1 số thực. In ra trị tuyệt đối của số đó. Cần xét khả năng người dùng không nhập ký số. Khi đó cần báo lỗi 'Not numeric'
- 3/. Cho người dùng nhập 1 số nguyên n . In ra bảng cửu chương của n . Nếu n nhập vào không phải là số nguyên thì báo lỗi và yêu cầu nhập lại.
- 4/. Cho người dùng nhập 2 số nguyên. In ra tổng hiệu tích thương của 2 số đó. Cần xét các khả năng gây lỗi.

 Gợi ý: sử dụng các exception `ValueError`, `ZeroDivisionError`, `Exception`

- 5/. Viết chương trình cho người dùng nhập số nguyên dương n , chương trình sẽ in ra n số ngẫu nhiên từ 0 đến 100. Nếu người dùng nhập kiểu dữ liệu khác (float, bool, ...) chương trình sẽ hiện ra câu 'Phải nhập số nguyên dương' và cho lặp lại việc nhập n . Chương trình sẽ lặp lại công việc trên nhiều lần và chỉ kết thúc khi người dùng nhập từ 'exit' hoặc nhập số 0 (zero).
- 6/. Cho người dùng nhập vào một số nguyên dương n thỏa lần lượt các yêu cầu sau đây. Nếu người dùng nhập vào số không đúng yêu cầu thì chương trình cho nhập lại
 - a. Điều kiện: $n > 0$. Nếu nhập đúng, chương trình in ra các số chẵn X ($0 \leq X \leq n$).
 - b. Điều kiện: $0 \leq n \leq 9$. Nếu nhập đúng, chương trình in ra cách đọc của số vừa nhập.
 - c. Điều kiện: $n < 10$ hoặc $n > 50$. Nếu nhập đúng, chương trình in ra n số ngẫu nhiên.
 - d. Điều kiện: n bội số của 5. Nếu nhập đúng, chương trình in ra bảng cửu chương 5.
 - e. Điều kiện: n nằm trong các số (2, 4, 6, 8, 10). Nếu nhập đúng, chương trình cho nhập tiếp 2 số nguyên i, j . Hãy tính và in ra cấp số cộng với số hạng đầu là i , công sai là j và số lượng phần tử của dãy là n .
- 7/. Viết chương trình cho nhập 2 số nguyên dương n và m sao cho số nhập sau (m) phải luôn lớn hơn số nhập trước (n). Nếu nhập đúng, in ra các số lẻ nằm trong khoảng từ n đến m . Khi người dùng nhập sai, thực hiện lần lượt các trường hợp sau:
 - a. Chỉ yêu cầu nhập lại m .
 - b. Yêu cầu nhập lại cả n và m .
- 8/. Cho nhập 2 số a, b sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho 7. Nếu nhập sai phải yêu cầu nhập lại cả 2 số cho đến khi đúng. Ngược lại, khi nhập đúng chương trình sẽ in ra các số chia hết cho 7 và nằm trong khoảng từ a đến b .
- 9/. Cho nhập 2 số nguyên dương a và b sao cho ước số chung lớn nhất của a và b phải là bội số của 5. Nếu nhập sai, cho nhập lại cả 2 số a và b . Nếu nhập đúng, đếm xem từ a đến b có bao nhiêu số chia hết cho 3.
- 10/. Nhập vào số nguyên dương x là lũy thừa của 2. Nếu nhập sai, yêu cầu nhập lại. Khi nhập đúng, in ra số x dưới dạng 2^k . Với k là số lượng số 2 tham gia vào phép nhân để có được giá trị x .

Ví dụ $x=32 = 2*2*2*2*2 \Rightarrow$ in ra 2^5

FILE - DIRECTORY - OPERATING SYSTEM

5.1. Text file

5.1.1. Đọc file

1/.

- Tạo file .txt: right click vào package, chọn New/File. Đặt tên file là tho.txt. Nội dung file như sau:

```
QUÊ HƯƠNG
...
Quê hương là chùm khế ngọt
Cho con trèo hái mỗi ngày
Quê hương là đường đi học
Con về rợp bướm vàng bay
...
```

- Viết chương trình đọc và hiển thị nội dung file trên ra màn hình console
- Viết chương trình đọc và hiển thị nội dung file trên ra màn hình console với số dòng cần đọc do người dùng nhập vào.

➤ Hướng dẫn: import islice trong module itertools để hỗ trợ đọc file theo dòng.

- Viết chương trình đọc và in n dòng cuối của file Que_Huong.txt ra màn hình. Với n do người dùng nhập từ bàn phím

➤ Hướng dẫn: import 2 module sys và os.

- Viết chương trình đọc nội dung file Que_Huong.txt và đưa vào 1 list. Sau đó in nội dung list L ra màn hình. Lần lượt thực hiện bằng 2 cách:

- Đọc cả file và đưa vào list 1 lần
- Đọc từng dòng đưa vào list cho đến khi hết file.

- Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về 1 list chứa những từ có chiều dài là dài nhất có trong file.

Minh họa kết quả thực hiện: ['HUONG', 'huong', 'huong', 'duong', 'Thach']

- Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về số dòng có trong file.

Minh họa kết quả thực hiện: Số lượng dòng có trong file: 8

- Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về tần suất xuất hiện của từng từ có trong file.

Minh họa kết quả thực hiện:

```
Tần suất xuất hiện của các từ: Counter({'...': 2, 'Que': 2, 'huong': 2, 'la': 2, 'QUE': 1, 'HUONG': 1, 'chum': 1, 'khe': 1, 'ngot': 1, 'Cho': 1, 'con': 1, 'treo': 1, 'hai': 1, 'moi': 1, 'ngay': 1, 'duong': 1, 'di': 1, 'hoc': 1, 'Con': 1, 've': 1, 'rop': 1, 'buom': 1, 'vang': 1, 'bay': 1, 'Giap': 1, 'Van': 1, 'Thach': 1})
```

➤ Hướng dẫn: import và sử dụng Counter (trong module collections).

- Viết chương trình đọc file text (txt) và đếm số lượng của mỗi ký tự có trong file. Xem như ký tự hoa và thường đều là ký tự thường.

Minh họa nội dung file abc.txt:

```
abc.txt -
German Unity Day
From Wikipedia, the free encyclopedia
The Day of German Unity (German: Tag der Deutschen Einheit) is the national day of Germany, celebrated on 3 October as a public holiday. It commemorates the anniversary of German reunification in 1990, when the goal of a united Germany that originated in the middle of the 19th century, was fulfilled again. Therefore, the
```

name addresses neither the re-union nor the union, but the unity of Germany. The Day of German Unity on 3 October has been the German national holiday since 1990, when the reunification was formally completed.

Minh họa kết quả thực hiện:

```
File Name: abc.txt
Counter({' ': 93, 'E': 64, 'N': 45, 'A': 42, 'T': 40, 'I': 36, 'O': 31, 'R': 29,
'H': 25, 'D': 19, 'M': 17, 'Y': 17, 'L': 15, 'F': 15, 'U': 14, 'C': 13, 'G': 13,
'S': 12, ',': 7, 'B': 6, 'W': 5, '9': 5, '.': 4, 'P': 4, '1': 3, '\n': 2, '0': 2,
'3': 2, ':': 1, '-': 1, 'K': 1, '(' : 1, ')': 1, 'V': 1})
```

5.1.2. Ghi file

- 8/. Viết chương trình ghi nội dung trích dẫn của bài thơ Quê hương vào file Que_Huong.txt. Sau đó, đọc nội dung file đã ghi ra màn hình.

☞ Hướng dẫn: import islice trong module itertools để hỗ trợ đọc file theo dòng.

5.1.3. Làm việc file chứa kiểu dữ liệu dạng số

- 9/. Viết chương trình thực hiện các yêu cầu sau, mỗi yêu cầu được viết thành 1 hàm riêng biệt. Viết chương trình chính lần lượt gọi các hàm đã viết. Giả sử file cần tạo có tên và đường dẫn là D:\Number.txt:
- Hàm *NhapN()*: Cho nhập số nguyên n , với nằm trong khoảng từ 2 đến 200. Nếu người dùng nhập sai (nhập ký tự không phải là các ký số, nhập số thực), chương trình sẽ báo lỗi và yêu cầu nhập lại. Khi người dùng nhập đúng, hàm trả về n .
 - Hàm *GhiFile(filename, n)*: hàm ghi n số nguyên được phát sinh ngẫu nhiên vào file có tên là filename. Giá trị ngẫu nhiên được phát sinh trong khoảng từ -9 đến +200.
 - Hàm *DocFile(filename)*: hàm đọc các giá trị có trong file filename và in ra màn hình.
 - Hàm *Tong(filename)*: hàm đọc các giá trị có trong file filename, hàm trả về tổng các số có trong file.
 - Hàm *LietKeSoNguyenTo(filename)*: hàm đọc các giá trị có trong file filename, in các số nguyên tố ra màn hình.
 - Hàm *LietKeSoChinhPhuong(filename)*: hàm đọc các giá trị có trong file filename, in các số chính phương ra màn hình. Với số chính phương (*square number hay perfect square*) là số có căn bậc hai là một số nguyên. Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3. Nếu trong file không có số chính phương thì in ra câu 'KHÔNG có số chính phương trong file'.
 - Hàm *LietKeSoHoanThien(filename)*: hàm đọc các giá trị có trong file filename, in các số hoàn thiện ra màn hình. Với số hoàn thiện (*Perfect number*) là số (n) có tổng các ước số không kể n bằng chính n . Ví dụ: 6 là số hoàn thiện vì $1+2+3=6$; hay số 28 vì $1+2+4+7+14=28$, ... Lưu ý: các số hoàn thiện nhỏ hơn 10 tỷ chỉ gồm các số: 6, 28, 496, 8.128, 33.550.336, 8.589.869. 056 .
 - Hàm *LietKeSoArmstrong(filename)*: hàm đọc các giá trị có trong file filename, in các số Armstrong ra màn hình. Với số Armstrong là số có đặc điểm sau: số đó gồm n ký số, tổng các lũy thừa bậc n của các ký số bằng chính số đó. Ví dụ 153 là một số có 3 ký số, và $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.

5.1.4. Lấy thông tin về file

- 10/. Viết chương trình cho nhập tên file, lấy đường dẫn tuyệt đối (absolute file path) của file.

Minh họa kết quả thực hiện:

```
Absolute file path test.txt: /home/students/path_fname
```

- 11/. Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về kích thước file tính theo byte.

Minh họa kết quả thực hiện: File size in bytes: 145

- a. Yêu cầu sử dụng phương thức stat trong module os
- b. Yêu cầu sử dụng getsize trong module os.path.

12/. Viết chương trình cho nhập tên file, chương trình trả về thông về ngày giờ tạo lập và ngày giờ hiệu chỉnh cuối cùng của file.

Minh họa kết quả thực hiện:

```
Last modified of test.txt file: Wed Apr 19 11:36:23 2017
Created of test.txt file: Wed Apr 19 11:36:23 2017
```

13/. Viết chương trình lấy thông tin các thuộc tính của file.

Minh họa kết quả thực hiện:

```
File : 8dbacd90-266d-11e7-a9c1-cf681af3cdf1.py
Access time : Fri Apr 21 14:06:03 2017
Modified time: Fri Apr 21 14:06:02 2017
Change time : Fri Apr 21 14:06:02 2017
Size : 304
```

14/. Viết chương trình tìm đường dẫn tuyệt đối của path; path là file hoặc thư mục hay là 1 liên kết (shortcut/link), path có tồn tại hay không? Và có tồn tại 1 liên kết nào đến path hay không?

Minh họa kết quả thực hiện:

```
File : 26cb6a20-266f-11e7-a9c1-cf681af3cdf1.py
Absolute : False
Is File? : True
Is Dir? : False
Is Link? : False
Exists? : True
Link Exists?: True
...
Is Dir? : False
Is Link? : False
Exists? : False
Link Exists?: False
```

5.2. CSV file

15/. Viết chương trình thực hiện các yêu cầu sau, trong đó, mỗi yêu cầu được viết riêng thành 1 hàm. Sau đó chương trình chính sẽ gọi các hàm này thực hiện:

- a. Hàm đọc file *Departments.csv* và in ra màn hình theo mẫu sau đây, trong đó:
 - Tên tiêu đề cột được in canh giữa 2 ký tự '|' ở 2 đầu
 - Field dạng chuỗi được in canh trái
 - Field dạng số được in canh phải.

Minh họa kết quả thực hiện:

DEPARTMENTS LIST				
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID	
10	Administration	200	1700	
20	Marketing	201	1800	
30	Purchasing	114	1700	
...				
...				
270	Payroll		1700	

Tổng cộng gồm 27 dòng dữ liệu

- b. Hàm đọc file *Departments.csv* và tạo mới 1 list L chứa những dòng sau:
 - Dòng tiêu đề của các cột có trong file CSV
 - Các dòng dữ liệu thỏa 1 trong 2 điều kiện:
 - location_id='1700' và department_name có chứa chuỗi 'ing'
 - Hoặc location_id='2400'

- Kết quả trả về là list vừa có
- c. Hàm ghi nội dung list L của câu b vào file có tên *KetQua.csv*. Với file *Departments.csv* do GV cung cấp, nội dung file *KetQua.csv* sẽ như sau:

```
department_id,department_name,manager_id,location_id
30,Purchasing,114,1700
40,Human Resources,203,2400
110,Accounting,205,1700
170,Manufacturing,,1700
190,Contracting,,1700
260,Recruiting,,1700
```

- d. Viết thêm hàm thực hiện các chức năng tương tự như câu a, chỉ khác là tên các cột khi in ra màn hình sẽ được đổi thành chữ in và thay các ký tự gạch dưới '_' (underscore) thành các khoảng trắng (space). Kết quả thực hiện sẽ có dạng

DEPARTMENTS LIST

DEPARTMENT ID	DEPARTMENT NAME	MANAGER ID	LOCATION ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
...			
...			
270	Payroll		1700

Tổng cộng gồm 27 dòng dữ liệu

16/.Viết chương trình để thao tác trên danh sách điểm thi của các thí sinh trong tập tin *danh sach.csv* (được GV cung cấp). Các thông tin trong *danh sach.csv* bao gồm : STT, Họ tên, ĐTB Môn 1, ĐTB Môn 2, ĐTB Môn 3 và Điểm ưu tiên.

- Yêu cầu:

- Xây dựng hàm *DocNoiDungFile_CSV(filename)* để đọc nội dung tập tin csv được truyền vào thông qua tham số *filename*.
 - Tính tổng điểm thi của từng thí sinh, biết rằng:

$$\text{Tổng điểm} = \text{ĐTB Môn 1} + \text{ĐTB Môn 2} + \text{ĐTB Môn 3} + \text{Điểm ưu tiên}.$$
 - Xử lý và trả về danh sách thí sinh đậu và danh sách thí sinh rớt. Trong đó thí sinh thi đậu nếu tổng điểm ≥ 20
- Xây dựng hàm *GhiNoiDungFile_CSV(filename, listcontent)* để ghi nội dung *listcontent* vào tập tin csv được khai báo thông qua *filename*
- Gọi hàm *DocNoiDungFile_CSV* để đọc danh sách điểm thi của các thí sinh dự thi, xuất danh sách thí sinh đậu và thí sinh rớt
- Gọi hàm *GhiNoiDungFile_CSV* để ghi danh sách thí sinh đậu và danh sách thí sinh rớt vào 2 tập tin csv là *DanhSachDau.csv* và *DanhSachRot.csv*

– Một số kết quả gọi ý:

Danh sách các thí sinh đậu (Tổng điểm ≥ 20):

Họ tên	Môn 1	Môn 2	Môn 3	Ưu tiên	Tổng
Đồng Quang An	7.0	6.9	7.1	0.0	21.0
Nguyễn Phước An	6.0	7.6	7.4	1.0	22.0
Nguyễn Thị Thanh An	6.3	6.8	6.3	1.0	20.4
Nguyễn Lê Hoàng Ân	8.0	7.7	7.5	0.5	23.7
Đỗ Phương Anh	6.6	6.2	6.2	1.0	20.0
Lê Quỳnh Anh	5.5	8.9	7.3	0.5	22.2
Lê Thị Kim Anh	7.0	8.0	6.7	1.0	22.7
Lê Huy Bảo	7.0	6.7	6.4	1.5	21.6
Trần Minh Cường	8.9	8.0	8.6	1.5	27.0

Danh sách các thí sinh rớt: (Tổng điểm < 20):

Họ tên	Môn 1	Môn 2	Môn 3	Ưu tiên	Tổng
Lê Văn Ân	6.1	5.3	6.5	1.5	19.4
Nguyễn Thị Nhật Anh	5.4	6.2	6.9	0.5	19.0
Hoàng Thị Ngọc Ánh	5.7	6.3	6.2	0.5	18.7
Nguyễn Thị Ánh	6.7	6.8	4.8	1.0	19.3
Huỳnh Anh Bằng	6.8	6.3	5.3	0.5	18.9
Nguyễn Hữu Cường	4.9	6.6	7.3	0.5	19.3

Đã ghi nội dung vào tập tin: DanhSachDau.csv

Đã ghi nội dung vào tập tin: DanhSachRot.csv

17/.Viết chương trình gồm 2 hàm sau, với tên file cần tạo mới và cần đọc là Countries.csv:

- Tao_File(TenFile): trong hàm sẽ thực hiện các công việc sau:

- Khai báo 1 list chứa nội dung các dòng có trong bảng sau.

country_id	country_name	region_id
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2

- Tạo file csv với tên là tham số TenFile được truyền cho hàm. Nội dung của file lấy trong list vừa khai báo ở trên và dấu phân cách được dùng là ký tự dấu thăng ('#' - hash).
- Yêu cầu: Viết 2 hàm Tao_File1 và Tao_File2, trong đó, hàm Tao_File1 ghi toàn bộ list chỉ trong 1 lệnh và hàm Tao_File2 thực hiện ghi mỗi lần một dòng.

- DocFile(TenFile): đọc nội dung file TenFile và in ra kết quả như hình minh họa.

country_id	country_name	region_id
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2

Tổng cộng gồm 5 dòng dữ liệu

Nội dung file csv sử dụng trong phần bài tập

<i>Departments.csv</i>	<i>DanhSach.csv</i>
department_id,department_name,manager_id,location_id	STT,Họ tên,ĐTB Môn 1,ĐTB Môn
10,Administration,200,1700	2,ĐTB Môn 3,Điểm ưu tiên
20,Marketing,201,1800	1,Đồng Quang An,7,6.9,7.1,0
30,Purchasing,114,1700	2,Nguyễn Phước An,6,7.6,7.4,1
40,Human Resources,203,2400	3,Nguyễn Thị Thanh
50,Shipping,121,1500	An,6.3,6.8,6.3,1
60,IT,103,1400	4,Lê Văn Ân,6.1,5.3,6.5,1.5
70,Public Relations,204,2700	5,Nguyễn Lê Hoàng
80,Sales,145,2500	Ân,8,7.7,7.5,0.5
90,Executive,100,1700	6,Đỗ Phương Anh,6.6,6.2,6.2,1
100,Finance,108,1700	7,Lê Quỳnh Anh,5.5,8.9,7.3,0.5
110,Accounting,205,1700	8,Lê Thị Kim Anh,7,8,6.7,1
120,Treasury,,1700	9,Nguyễn Thị Nhật
130,Corporate Tax,,1700	Anh,5.4,6.2,6.9,0.5
140,Control And Credit,,1700	10,Hoàng Thị Ngọc
150,Shareholder Services,,1700	Ánh,5.7,6.3,6.2,0.5
160,Benefits,,1700	11,Nguyễn Thị Ánh,6.7,6.8,4.8,1
170,Manufacturing,,1700	12,Huỳnh Anh
180,Construction,,1700	Bằng,6.8,6.3,5.3,0.5
190,Contracting,,1700	13,Lê Huy Bảo,7,6.7,6.4,1.5
200,Operations,,1700	14,Nguyễn Hữu
210,IT Support,,1700	Cường,4.9,6.6,7.3,0.5
220,NOC,,1700	15,Trần Minh
230,IT Helpdesk,,1700	Cường,8.9,8,8.6,1.5
240,Government Sales,,1700	
250,Retail Sales,,1700	
260,Recruiting,,1700	
270,Payroll,,1700	

5.3. Directory (Thư mục)

18/ Viết chương trình cho nhập tên file kèm ký tự đại diện, thực hiện sắp xếp danh sách các file theo ngày giờ tạo lập.

Minh họa kết quả thực hiện: giả sử yêu cầu liệt kê các file **"*.txt"** có trong thư mục hiện tại:

```
result.txt
temp.txt
myfile.txt
mynewtest.txt
mytest.txt
abc.txt
test.txt
```

19/ Viết chương trình cho nhập tên thư mục (myDict), chương trình sẽ liệt kê các thư mục con có trong myDict được sắp xếp theo thứ tự tăng dần của ngày tạo lập.

20/ Viết chương trình cho nhập tên (đặt là path) thư mục hoặc tên file hay tên đặc biệt khác (như socket, FIFO, device file, ...). Chương trình thực hiện kiểm tra xem path là thư mục, tập tin hoặc tên đặc biệt khác.

➤ Hướng dẫn: sử dụng các phương thức `os.path.isdir` và `os.path.isfile` để kiểm tra

Ví dụ, với file có tên `abc.txt`, khi thực hiện kiểm tra sẽ cho kết quả là: `It is a normal file`

21/.Viết chương trình để tạo danh sách chứa các tập tin có trong thư mục hiện hành bằng ký tự đại diện.

22/.Viết chương trình cho biết đường dẫn thư mục của người dùng (home directory).

Minh họa kết quả thực hiện: C:\Users\Administrator

23/.Viết chương trình cho nhập tên folder cùng đường dẫn (tạm đặt là path), in ra danh sách các file có trong folder đó.

Minh họa kết quả thực hiện: khi path= '/home/students'

Sẽ in ra: ['test.txt', '.mysql_history', '.bash_logout', '.bash_history', '.profile', 'abc.py', '.viminfo', 'mynewtest.txt', 'myfile.txt', 'logging_example.out', '.web-term.json', 'abc.txt', '64a57280-272f-11e7-9ce4-832a8e030fef.py', 'exercise.cs', '.bashrc', 'Example.cs', 'myfig.png', 'file.out', 'line.gif', 'mmm.txt\n', 'temp.txt', 'dddd.txt\n', 'sss.dat\n', 'result.txt', 'output.jpg', '26492-1274250701.png', 'mytest.txt']

5.4. Khác

24/.Viết chương trình thu thập các đối số của dòng lệnh, sau đó cho biết tên của script, số lượng đối số và danh sách các đối số có trong dòng lệnh.

Ví dụ, với script có tên test.py, khi thực hiện dòng lệnh sau:

```
prashanta@server:~$ python test.py arg1 arg2 arg3
```

Sẽ thu được kết quả là

```
This is the name/path of the script: test.py
('Number of arguments:', 4)
('Argument List:', ["'test.py'", 'arg1', 'arg2', 'arg3'])
```

25/.Viết chương trình cho nhập tên (đặt là path). Phân tách path thành 2 phần: phần 1 gồm tên file (kể cả đường dẫn nếu có), phần 2 gồm phần mở rộng của.

➤ Hướng dẫn: sử dụng các phương thức os.path.splitext()

Ví dụ, với các tên được nhập lần lượt là: ['test.txt', 'filename', '/user/system/test.txt', '/', '']

Sẽ cho kết quả:

```
"test.txt"           : ('test', '.txt')
"filename"           : ('filename', '')
"/user/system/test.txt" : ('/user/system/test', '.txt')
"/"                  : ('/', '')
""                   : ('', '')
```

26/.Viết một chương trình Python để lấy tên của máy chủ (hostname) mà chương trình đang chạy

➤ Hướng dẫn: sử dụng phương thức socket.gethostname () để lấy host name.

27/.Viết chương trình Python để kiểm tra chuỗi địa chỉ IP (IP address) có hợp lệ hay không? Ví dụ với AddrStr= '127.0.0.2561' ⇒ Invalid IP

➤ Hướng dẫn: sử dụng phương thức socket.inet_aton(AddrStr) để kiểm tra tính hợp lệ của địa chỉ IP của chuỗi AddrStr.

28/.Viết chương trình Python để kiểm tra operating system name, platform and platform release của máy tính đang dùng.

➤ Hướng dẫn: sử dụng các thuộc tính hoặc phương thức sau trong 2 module os và platform để lấy thông tin cần dùng:

```
os.name           ⇒ Operating system name
platform.system() ⇒ Platform name
platform.release() ⇒ Platform release
```

29/.Viết chương trình Python để kiểm tra xem python shell bạn đang dùng sử dụng ở mode 32 hay 64 bit.

↪ Hướng dẫn: sử dụng phương thức `struct.calcsize("P")` trong module `struct` để lấy thông tin cần dùng:

CLASS

6.1. Class & Object

1/. Tạo class `PhuongTrinhBac1` gồm:

- *Attribute*: `hesoA`, `hesoB`
- *Method*: (khởi tạo - `__init__`), `TinhNghiem`, `InNghiem`

2/. Tạo class `PhuongTrinhBac2` gồm:

- *Attribute*: `hesoA`, `hesoB`, `hesoC`
- *Method*: (khởi tạo - `__init__`), `TinhNghiem`, `InNghiem`

3/. Tạo class `cRectangle`, với 3 phương thức:

- (i)- *constructor* (khởi tạo - `__init__`): nhận 2 tham số là *length* và *width* (có kiểu dữ liệu là số).
Phương thức khởi tạo 2 thuộc tính của class là chiều dài $l=length$ và chiều rộng $w=width$.
- (ii)- *rectangle_area*: trả về diện tích hình chữ nhật.
- (iii)- *rectangle_perimeter*: trả về chu vi hình chữ nhật.

Yêu cầu:

- Liệt kê tất cả các namespace của class vừa xây dựng.
↳ Gợi ý: sử dụng `className.__dict__`
- Khởi tạo 1 *object* thuộc class `cRectangle` với kích thước chiều dài và chiều rộng do người dùng nhập vào. In ra:
 - Chu vi và diện tích của *object* vừa tạo
 - Tên của *class* mà *object* đang là một thể hiện (*instance*). Gợi ý: sử dụng `type(objectName).__name__`
 - Hiển thị giá trị từng thuộc tính của *object* đang có. Gợi ý: sử dụng `className.__dict__`

4/. Tạo class `cCircle`, với 3 phương thức:

- (i)- *constructor* (khởi tạo - `__init__`): nhận tham số r là bán kính (có kiểu dữ liệu là số).
Phương thức khởi tạo thuộc tính của class là $radius = r$.
- (ii)- *circle_area*: trả về diện tích hình tròn.
- (iii)- *circle_perimeter*: trả về chu vi hình tròn.

Yêu cầu: khởi tạo 1 *object* thuộc class `cCircle` với bán kính do người dùng nhập vào. In ra chu vi và diện tích của *object* vừa tạo và tên của *class* mà *object* đang là một thể hiện (*instance*).

5/. Tạo class `cString`,

a. Tạo class `cString` với 3 phương thức:

- (i)- *constructor* (khởi tạo - `__init__`): khởi tạo thuộc tính S là chuỗi rỗng.
- (ii)- *get_String*: cho người dùng nhập chuỗi rồi gán cho thuộc tính S .
- (iii)- *print_String*: in giá trị của thuộc tính S dưới dạng chữ in hoa.

Ví dụ: Nhập chuỗi: sài gòn # do phương thức `get_String` thực hiện
SÀI GÒN # do phương thức `print_String` thực hiện

b. Mở rộng class `cString`, trong đó có phương thức `sub_sets` nhận tham số là 1 list $L1$ chứa các phần tử. Phương thức trả về 1 list $L2$ chứa tất cả các tổ hợp được tạo thành từ các phần tử có trong $L1$.

Ví dụ: $[4, 5] \Rightarrow [[], [5], [4], [4, 5]]$

$['A', 'B', 'C'] \Rightarrow [[], ['C'], ['B'], ['B', 'C'], ['A'], ['A', 'C'], ['A', 'B'], ['A', 'B', 'C']]$

- c. Tiếp tục mở rộng class *cString* bằng cách thêm phương thức *reverse_words* nhận tham số là 1 chuỗi (S). Phương thức trả về 1 chuỗi mới bằng cách đảo ngược các từ có trong S.

Ví dụ: Chuỗi ban đầu: **Thành phố Hồ Chí Minh**

Chuỗi sau khi đảo ngược các từ: **Minh Chí Hồ phố Thành**

6/. Xây dựng class *cNumber*

- a. Xây dựng class *cNumber* gồm 2 phương thức:

(i)- *Int_To_Roman (Int_Para)*: để chuyển đổi số nguyên (hệ thập phân) sang chữ số La Mã (*Roman numeral*). Kết quả trả về của phương thức là dữ liệu kiểu chuỗi.

Ví dụ: 127 = CXXVII

300 = CCC

3321 = MMMCCCXXI

(ii)- *Roman_To_Int (string_Para)*: để chuyển đổi số La Mã sang số nguyên hệ thập phân. Kết quả trả về của phương thức là số nguyên.

Ví dụ: MMMCMLXXXVI = 3986

CMLIXIV = 963

CDXCXL = 530

➤ Nhắc lại về chữ số La mã (theo wikipedia)

- Cách viết:

- Có bảy chữ số La Mã cơ bản:

Ký tự	M	D	C	L	X	V	I
Giá trị	1000	500	100	50	10	5	1

- I chỉ có thể đứng trước V hoặc X, X chỉ có thể đứng trước L hoặc C, C chỉ có thể đứng trước D hoặc M.
- Nhiều ký hiệu có thể được kết hợp lại với nhau để chỉ các số với các giá trị khác chúng. Thông thường người ta quy định (i):
 - Các chữ số I, X, C, M, không được lặp lại quá ba lần liên tiếp;
 - Các chữ số V, L, D không được lặp lại liên tiếp. Chính vì thế mà có 6 nhóm chữ số đặc biệt được nêu ra trong bảng sau:

Ký tự	CM	CD	XC	XL	IX	IV
Giá trị	900	400	90	40	9	4

- Đối với những số lớn hơn (4000 trở lên), một dấu gạch ngang được đặt trên đầu số gốc để chỉ phép nhân cho 1000.

Ký tự	\bar{V}	\bar{X}	\bar{L}	\bar{C}	\bar{D}	\bar{M}
Giá trị	5.000	10.000	50.000	100.000	500.000	1.000.000

Do các ký tự vừa liệt kê không có sẵn trong Python, vì vậy bài tập này bỏ qua các trường hợp quy định trong phần (i).

- Cách tính: Tính từ trái sang phải giá trị của các chữ số và nhóm chữ số giảm dần.

Ví dụ: CVI = 106

CMIV = 904

MMMM = 4000

- b. Mở rộng class *cNumber* bằng cách thêm phương thức *twoSum* nhận tham số là một list (L) chứa các số nguyên và số nguyên (*target*). Phương thức tìm và trả về 2 số nguyên đang có trong L sao cho tổng giá trị của 2 số này = *target*.

Ví dụ: L = [10, 20, 10, 40, 50, 60, 70] và target = 100

Kết quả sẽ in ra: $40 + 60 = 100$

- c. Tiếp tục mở rộng *class cNumber* bằng cách thêm phương thức *is_valid_parenthese* giúp kiểm tra tính hợp lệ của một chuỗi chỉ chứa các dấu ngoặc đơn, '(', ')', '{', '}', '[', ']' và ']'. Với yêu cầu các dấu ngoặc này phải được đóng theo đúng thứ tự,

Ví dụ "`{ () [] } "`", "`{ ([]) } "`" và "`() { } [] "`" là hợp lệ
nhưng "`() [{ }] "`", "`[] "`", "`{ ([]) }`" và "`{ { { }`" là không hợp lệ.

🔗 Gợi ý:

- Sử dụng *dictionary* chứa các cặp ngoặc mỗi bộ *key:value* là 1 cặp ngoặc.
- Sử dụng *list* làm *stack*. Khi duyệt hết chuỗi, nếu *stack* rỗng sẽ trả về *True*, ngược lại trả về *False*.

- d. Tiếp tục mở rộng *class cNumber* bằng cách thêm phương thức *threeSum* nhận tham số là 1 *list* chứa các số nguyên với giá trị âm dương tùy ý. Phương thức sẽ trả về tất cả các bộ (không trùng nhau) chứa 3 số có trong *list* sao cho tổng của chúng = zero (0).

Ví dụ: List ban đầu: [-25, -10, -7, -3, 2, 4, 8, 10]
List kết quả: [[-10, 2, 8], [-7, -3, 10]]
List ban đầu: [-25, -10, -7, -3]
Không tìm thấy 3 số có tổng = 0

- e. Tiếp tục mở rộng *class cNumber* bằng cách thêm phương thức *pow* nhận tham số là 2 số nguyên *x* và *n*. Phương thức trả về kết quả là x^n (*x* lũy thừa *n*).

Ví dụ:

<i>x</i>	<i>n</i>	Xuất ra kết quả
2	4	$2^4 = 16$
2	-3	$2^{-3} = 0.125$
3	5	$3^5 = 243$
100	0	$100^0 = 1$

7/. Xây dựng *class PhanSo* (phân số).

- Viết các phương thức khởi tạo, nhập 1 phân số, xuất 1 phân số, cộng, trừ, nhân, chia, so sánh 2 phân số, hàm rút gọn 1 phân số.
- Tạo ra 1 *list L* chứa *n* phân số với *n* là số nguyên do người dùng nhập vào. Giá trị của tử và mẫu số được phát sinh ngẫu nhiên từ -10 đến +10.
- Tìm phân số lớn nhất, phân số nhỏ nhất, tổng các phân số có trong *list L*.
- Sắp xếp các phân số giảm dần theo giá trị.

6.2. Kế thừa (Inheritance)

8/. Xây dựng *class HonSo* (Hỗn số).

- Viết các phương thức khởi tạo, nhập 1 hỗn số, xuất 1 hỗn số, cộng, trừ, nhân, chia, so sánh 2 hỗn số, hàm rút gọn 1 hỗn số.
- Tạo ra 1 *list L* chứa *n* hỗn số với *n* là số nguyên do người dùng nhập vào. Giá trị của phần nguyên, tử và mẫu số được phát sinh ngẫu nhiên từ -10 đến +10.
- Tìm hỗn số lớn nhất, hỗn số nhỏ nhất, tổng các hỗn số có trong *list L*.
- Sắp xếp các hỗn số giảm dần theo giá trị.

9/. Viết chương trình quản lý sách trong 1 thư viện, với các yêu cầu sau:

- Thông tin về sách gồm:
 - Sách giáo khoa: mã sách, tên sách, ngày nhập (ngày/tháng/năm), đơn giá bìa, số lượng, nhà xuất bản, tình trạng (mới/cũ tính theo phần trăm)

- Sách tham khảo: mã sách, tên sách, ngày nhập (ngày/tháng/năm), đơn giá bìa, số lượng, nhà xuất bản, tỷ lệ thuế (% , từ 1-20%)
- Tạo class *Sach* với các thuộc tính và phương thức chung.
- Tạo class *SachGiaoKhoa* và *SachThamKhao* kế thừa từ class *Sach* với các thuộc tính riêng và phương thức cần thiết. Đối với class *SachGiaoKhoa*: tiền đền bù khi mất sách là đơn giá bìa * % tình trạng mới/cũ. Đối với class *SachThamKhao*: tiền đền bù khi mất sách là đơn giá bìa * thuế.
- Viết chương trình:
 - Nhập xuất danh sách các loại sách.
 - Trong sách giáo khoa có bao nhiêu sách cũ (<100%) và bao nhiêu sách mới 100%.

6.3. Abstract

10/. Viết chương trình tính chu vi và diện tích của 1 số hình như sau: Hình tròn, hình chữ nhật, hình tam giác, với các yêu cầu

- Xây dựng abstract class *Hình* chứa 2 phương thức trừu tượng *TinhChuVi* và *TinhDienTich*
- Xây dựng class *HinhTron* kế thừa từ class *Hình* với 1 thuộc tính *BanKinh* và các phương thức *__init__*, *TinhChuVi* và *TinhDienTich*
- Xây dựng class *HinhChuNhat* kế thừa từ class *Hình* với 2 thuộc tính *Dai* và *Rong* cùng các phương thức *__init__*, *TinhChuVi* và *TinhDienTich*
- Xây dựng class *HinhTamGiac* kế thừa từ class *Hình* với 3 thuộc tính *a*, *b*, *c* cùng các phương thức *__init__*, *TinhChuVi* và *TinhDienTich*. Trong đó cần kiểm tra tính hợp lệ của tam giác.

☞ **Nhắc lại:**

- Công thức tính diện tích $s = \sqrt{p(p-a)(p-b)(p-c)}$
- Công thức tính các đường cao: $h_a = 2s/a$, $h_b = 2s/b$, $h_c = 2s/c$.

(Với p là nửa chu vi của tam giác).

DATA STRUCTER

7.1. Enum

- 1/. Viết chương trình khởi tạo và gán giá trị cho enum object tên là Country gồm các thành phần sau:

```
class Country(Enum):
    India = 355
    Andorra = 376
    Albania = 355
    USA = 213
    Afghanistan = 93
    Angola = 244
    Algeria = 213
```

Yêu cầu:

- In giá trị name và data của thành phần “Albania”.
- In giá trị name và data của tất cả các thành phần có trong object Country (Lưu ý: các thành phần có value trùng nhau và được khai báo sau trong enum sẽ không được hiển thị).
- In giá trị của tất cả các thành phần có trong object Country theo thứ tự tăng dần của name.
- In giá trị của tất cả các thành phần có trong object Country theo thứ tự tăng dần của value.

7.2. Array

- 2/. Viết chương trình tạo 1 array chứa 6 số nguyên.
- In các giá trị có trong array ra màn hình.
 - Sử dụng phương thức `buffer_info` để lấy thông tin về bộ đệm mảng bao gồm địa chỉ bắt đầu bộ đệm mảng trong bộ nhớ và số lượng phần tử của mảng.
 - Sử dụng hàm `len` để lấy số lượng phần tử của mảng.

7.3. Search & sort in list

- Viết hàm để sắp xếp danh sách các phần tử bằng thuật toán Bubble sort.
- Viết hàm để sắp xếp danh sách các phần tử bằng thuật toán Selection sort.
- Viết hàm để sắp xếp danh sách các phần tử bằng thuật toán Insertion sort.
- Viết hàm để sắp xếp danh sách các phần tử bằng thuật toán Quick sort.
- Trong khoa học máy tính, thuật toán tìm kiếm tuần tự (hay tìm kiếm tuyến tính) thực hiện tìm kiếm một giá trị trong một mảng bằng cách lần lượt so sánh các phần tử trong mảng với giá trị cần tìm cho đến khi tìm thấy hoặc đã tìm hết trên mảng. Viết hàm thực hiện tìm kiếm phần tử X trên 1 danh sách
- Trong khoa học máy tính, thuật toán tìm kiếm nhị phân (binary search) thực hiện tìm kiếm một giá trị trong một mảng đã được sắp xếp. Viết hàm nhận 2 tham số là list1 chứa các số nguyên đã được sắp xếp và 1 số nguyên x, hàm thực hiện tìm kiếm giá trị x trên list1 bằng thuật toán tìm kiếm nhị phân. Hàm trả về kết quả True nếu tìm thấy và trả về False nếu không tìm thấy. Hãy cài đặt hàm này

Ví dụ cần tìm x=45 trên list1=[6, 12, 17, 23, 38, 45, 77, 84, 90] => trả về True

7.4. Queue & Stack

7.4.1. queue

- 9/. Viết chương trình thực hiện các công việc sau:
 - a. Lần lượt tạo ra 5 số ngẫu nhiên rồi đưa vào *FIFO queue*. Đối với mỗi số ngẫu nhiên x được tạo ra, chương trình cần in x ra màn hình trước khi đưa x vào *queue*.
 - b. Cho biết kích thước của *queue* sau khi đưa 5 số vào.
 - c. Sử dụng lệnh *for* và phương thức *queueName.get_nowait* để lần lượt lấy từng số ra và in ra màn hình, đồng thời in số lượng phần tử còn lại sau mỗi lần lấy ra.
 - d. Thực hiện tương tự câu c nhưng sử dụng lệnh *while not queueName.empty()*.
- 10/. Thực hiện tương tự bài tập 9 với loại *queue* sử dụng là *LIFO queue*
- 11/. Thực hiện tương tự bài tập 9 với loại *queue* sử dụng là *Priority queue*

7.4.2. heapq

- 12/. Cho 1 danh sách chứa số nguyên với số lượng và giá trị tùy ý. Sử dụng *heapq* để đưa các giá trị này vào *heap*.
- 13/. Cho 1 danh sách chứa số nguyên với số lượng và giá trị tùy ý. Xuất các số trong list ra theo thứ tự tăng dần. Yêu cầu: không viết chương trình thực hiện sắp xếp list trên.
- 14/. Viết chương trình tìm ba số nguyên lớn nhất từ một danh sách các số đã cho bằng thuật toán *Heap queue*.

Ví dụ: với list = [25, 35, 22, 85, 14, 65, 75, 22, 58]

Chương trình in ra: Ba số lớn nhất trong list là: [85, 75, 65]

Hai số nhỏ nhất trong list là: [14, 22]

- 15/. Viết chương trình tìm giá trị nhỏ thứ k trong danh sách các số nguyên chưa có thứ tự.
- 16/. Viết chương trình triển khai một *heapsort* bằng cách đẩy tất cả các giá trị của list đang có lên một *heap* và sau đó lần lượt lấy các phần tử ra.
- 17/. Sử dụng module *heapq*, lần lượt thực hiện các yêu cầu sau:
 - a. Đưa 4 tuple sau vào *myHeapQueue*: ('V',11), ('V',14), ('V',17), ('V',11), ('V',13), ('V',19), ('V',12). In các giá trị trong *myHeapQueue* ra màn hình
 - b. Sử dụng hàm ***heappushpop*** để thêm phần tử ('V',6) vào lấy ra 1 phần tử từ *myHeapQueue*. In kết quả thực hiện ra màn hình và nhận xét.
 - c. Sử dụng hàm ***heapreplace*** để thêm phần tử ('V',2) vào lấy ra 1 phần tử từ *myHeapQueue*. In kết quả thực hiện ra màn hình và nhận xét.
 - d. Sử dụng hàm ***nLargest*** để lấy ra 3 phần tử có giá trị lớn nhất.
 - e. Sử dụng hàm ***nsmallest*** để lấy ra 2 phần tử có giá trị nhỏ nhất.

7.4.3. bisect

- 18/. Cho 1 list chứa các số nguyên đã được sắp tăng dần (ví dụ: lst = [1,2,4,5]). Sử dụng module *bisect*, lần lượt thực hiện các yêu cầu sau:
 - a. Viết hàm *indexLeft* nhận 2 tham số, tham số 1 là 1 list (*listName*), tham số 2 là giá trị cần chèn vào list. Hàm sử dụng phương thức *bisect.bisect_left(listName, value)* để trả về vị trí của *value* nếu được chèn vào *listName*. Lần lượt thực hiện với các *value* lần lượt là 6, 3 và 2.
 - b. Viết hàm *indexRight*, thực hiện tương tự như câu a, chỉ khác là trong đó sử dụng phương thức *bisect.bisect_right(listName, value)*. So sánh kết quả thực hiện của 2 câu a và câu b.
- 19/. Cho 1 list chứa các số nguyên chưa được sắp tăng dần (ví dụ: lst1 = [25, 45, 36, 47, 69, 48, 68, 78, 14, 36]). Tạo 1 list thứ 2 (*lst2*) bằng cách sử dụng module *bisect*, trong đó dùng các

phương thức `bisect.bisect(listName, value)` để xác định vị trí của `value` nếu được chèn vào `listName` để các giá trị trong `listName` có giá trị tăng dần và phương thức `bisect.insort(listName, value)` để chính thức chèn `value` vào `list2`. In `list2` ra màn hình.

20/. Viết chương trình cho người dùng nhập vào 1 số nguyên dương thuộc cơ số thập phân X. Chương trình thực hiện việc đổi số nguyên X sang các cơ số khác (2, 8, 16).

21/. Cho người dùng nhập 1 chuỗi S, sử dụng `stack` để đảo ngược 1 chuỗi S.

Ví dụ ABLE WAS I ERE I SAW ELBA => ABLE WAS I ERE I SAW ELBA

22/. Cho nhập biểu thức dạng `Infix`.

- Sử dụng `stack` để chuyển biểu thức vừa nhập sang dạng `Postfix`.
- Thực hiện tính toán giá trị cho biểu thức `Postfix` vừa có.

23/. Lần lượt viết chương trình quản lý 1 `Queue` bằng cách sử dụng danh sách kê và `DSLK` đơn với cấu trúc dữ liệu như sau

```
struct QUEUE
{ char *qArray;
  int qMax;
  int qFront;
  int qRear;
};
```

```
struct NODE
{ int data;
  NODE *pNext;
};
```

```
struct QUEUE
{ NODE * qFront;
  NODE * qRear;
};
```

- Tạo 1 queue gồm n node, với n do người dùng nhập vào.
- Để tiện cho việc cài đặt các chức năng khác, viết hàm `ShowQueue` để liệt kê các giá trị đang có queue
- Thêm node vào danh sách (hàm `EnQueue`). Trong đó, nếu sử dụng cấu trúc dạng danh sách kê thì hàm sẽ thực hiện dồn mảng về đầu ngay sau khi phần tử cuối được dùng (`q.qRear=qMax`)
- Lấy 1 node ra khỏi danh sách (hàm `DeQueue`).
- Cho nhập số lượng node cần lấy ra (k), nếu còn đủ k node thì thực hiện lấy ra, ngược lại thông báo không đủ k node trong queue.

24/. Viết chương trình thực hiện yêu cầu sau:

- Khai báo cấu trúc `Queue`
- Sau đó khai báo các biến sau: `QUEUE q1, q2, q3, q4, q5;`
- Viết các hàm sau:
 - `void Tach(QUEUE q, QUEUE &q4, QUEUE &q5)`: Tách `q1` thành 2 queue, sao cho:
 - Queue thứ nhất (`q4`) chứa các phần tử là số nguyên tố.
 - Queue thứ hai (`q5`) chứa các phần tử còn lại.
 - `void Nhap(QUEUE q1, QUEUE q2, QUEUE &q3)`: Giả sử `q1` và `q2` đã được sắp tăng dần. Nối `q1` và `q2` thành `q3` sao cho `q3` vẫn có thứ tự tăng dần.

7.5. Linked list

25/. Tạo danh sách liên kết với trường info có kiểu dữ liệu là số nguyên.

- Viết giải thuật thêm phần tử có nội dung x vào danh sách liên kết có thứ tự tăng dần sao cho sau khi thêm danh sách liên kết vẫn có thứ tự tăng.
- Loại bỏ phần tử có nội dung là x trong danh sách liên kết có thứ tự tăng dần.
- Cho 2 danh sách liên kết `pHead1`, `pHead2` có thứ tự tăng dần theo info. Viết giải thuật Merge để trộn 2 danh sách liên kết này lại sao cho danh sách liên kết sau khi trộn cũng có thứ tự tăng dần.

- 26/. Viết chương trình tạo một danh sách liên kết theo giải thuật thêm vào đầu danh sách, mỗi node chứa một số nguyên.
- Viết hàm tên `Delete_Node` để xóa node có địa chỉ p.
 - Viết một hàm loại bỏ tất cả các node có nội dung x trong danh sách liên kết pHead.
 - Viết hàm `Copy_List` trên danh sách liên kết để tạo ra một danh sách liên kết mới giống danh sách liên kết cũ.
 - Ghép một danh sách liên kết có địa chỉ đầu là pHead2 vào một danh sách liên kết có địa chỉ đầu là pHead1 ngay sau phần tử thứ i trong danh sách liên kết pHead1.
 - Viết hàm lọc danh sách liên kết để tránh trường hợp các node trong danh sách liên kết bị trùng info.
 - Đảo ngược vùng liên kết của một danh sách liên kết sao cho:
 - pHead sẽ chỉ đến phần tử cuối
 - Phần tử đầu có liên kết là NULL.
 - Viết hàm `Left_Traverse (NodePointer pHead)` để duyệt ngược danh sách liên kết.
 - Viết giải thuật tách một danh sách liên kết thành hai danh sách liên kết, trong đó một danh sách liên kết chứa các phần tử có số thứ tự lẻ và một danh sách liên kết chứa các phần tử có số thứ tự chẵn trong danh sách liên kết cũ.
- 27/. Cho 2 xâu liên kết L1 và L2. Giả thiết mỗi phần tử của chúng chỉ có 2 thông tin :
- Key : kiểu Integer
 - Next : con trỏ chỉ đến phần tử kế
- Viết chương trình tạo một xâu liên kết L nối từ 2 xâu L1 và L2 sao cho :
 - Các phần tử trong L có giá trị tăng
 - Không có trường hợp trùng nhau
 - Viết chương trình tách xâu l thành 2 xâu L3 và L4, trong đó L3 chứa các phần tử có khóa >0 và L4 chứa các phần tử có khóa <0 .
 - Đánh giá chi phí thuật toán của 2 câu a và b ở trên.

7.6. Binary search tree

7.6.1. Nhập các phần tử vào cây NPTK theo các cách sau

- Nhập trực tiếp từ bàn phím
- Phát sinh ngẫu nhiên.
- Nhập từ mảng 1 chiều
- Nhập từ 1 danh sách liên kết

7.6.2. Xuất

- Xuất toàn bộ cây theo 3 phương pháp NLR, LNR và LRN
- Cài đặt thuật toán duyệt cây nhị phân (NLR) không dùng đệ quy (dùng stack)
- Cài đặt thuật toán duyệt cây nhị phân theo mức không dùng đệ quy (dùng queue)

7.6.3. Liệt kê

- Các node lá
- Các node có 1 cây con
- Các node có 2 cây con.
- Các node lá có khóa $< x$ (x là tham số đầu vào của hàm). Thực hiện tương tự cho hàm đếm số node có khóa $> x$.

7.6.4. Tìm kiếm

39/. Tìm x có trên cây hay không? (x là tham số đầu vào của hàm)

7.6.5. Tổng - Trung bình

40/. Các node có trên cây.

41/. Các node lá

42/. Các node có 1 cây con

43/. Các node có 2 cây con.

44/. Các node có giá trị là số lẻ (tương tự cho trường hợp số chẵn).

45/. Các node lá có khoá $< x$ (x là tham số đầu vào của hàm). Thực hiện tương tự cho hàm đếm số node có khoá $> x$

46/. Viết thủ tục/hàm để tính tổng số node có 1 nhánh con (con trái HAY con phải) bằng cách dùng thuật toán duyệt gốc giữa NLR.

47/. Tìm giá trị nhỏ nhất trên cây

7.6.6. Đếm

48/. Tổng số node có trên cây.

49/. Số node lá (node bậc 0)

50/. Số node có 1 cây con, không phân biệt cây con trái hoặc cây con phải (node bậc 1)

51/. Số node chỉ có 1 cây con phải

52/. Số node có 1 cây con trái

53/. Số node có 2 cây con. (node bậc 2)

54/. Số node có giá trị là số lẻ (tương tự cho trường hợp số chẵn).

55/. Số node lá có khoá $< x$ (x là tham số đầu vào của hàm). Thực hiện tương tự cho hàm đếm số node có khoá $> x$

56/. Hãy viết một hàm không đệ quy đếm số phần tử trong cây.

57/. Hãy viết một hàm đệ quy đếm số phần tử trong cây.

58/. Số node trên từng mức của cây

59/. Viết hàm đếm số node trong cây mà không dùng đệ quy. Hướng dẫn:.. Khử đệ quy bằng cách dùng stack S.

THAM KHẢO

- [1]. <https://www.w3resource.com/python-exercises/>
- [2]. https://www.w3schools.com/python/python_exercises.asp
- [3]. <https://www.programiz.com/python-programming>

Trong quá trình học tập, sinh viên có thể tham khảo thêm các link có bài tập và bài giải sau:

- [1]. [Python Exercises, Practice, Solution - w3resource](#)
- [2]. <https://quantrimang.com/hon-100-bai-tap-python-co-loi-giai-code-mau-142456>
- [3]. <https://kinhnghiemtinhoc.com/hon-100-bai-tap-python-co-loi-giai-code-mau/>